JSC-12743
Revision A
E80-10175
NASA CR-
166672

AS-BUILT DESIGN
FOR
LACIE PHASE III AUTOMATIC STATUS AND TRACKING SYSTEM

Job Order 71-695

(E80-10175) AS-BUILT DESIGN SPECIFICATION FOR LACIE PHASE 3 AUTOMATIC STATUS AND TRACKING SYSTEM (Lockheed Electronics Co.) 158 p HC A08/MF A01 CSCL 05B

N80-28776

Unclas
G3/43 00175

Prepared By

Lockheed Electronics Company, Inc.
System and Services Division
Houston, Texas

Contract NAS 9-15200

For

EARTH OBSERVATIONS DIVISION

SCIENCE AND APPLICATIONS DIRECTORATE

National Aeronautics and Space Administration

LYNDON B. JOHNSON SPACE CENTER

Houston, Texas

June 1977

LEC-10419
Revision A

JSC-12743
Revision A

AS-BUILT DESIGN SPECIFICATION
FOR
LACIE PHASE III AUTOMATIC
STATUS AND TRACKING SYSTEM

Job Order 71-695

Prepared By

D. L. Smith
J. L. Allison
J. M. Everette
C. C. deValcourt

APPROVED BY

LEC

P. L. Krumm, Supervisor
Applications Software Section

NASA

V. M. Dauphin, Data Manager
System and Facilities Branch

Prepared By

Lockheed Electronics Company, Inc.

For

Earth Observation Division
Space and Life Sciences Directorate

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LYNDON B. JOHNSON SPACE CENTER
HOUSTON, TEXAS

June 1977

LEC-10419
Revision A

CONTENTS

FIGURES

TABLES

1. INTRODUCTION

1.1 PURPOSE AND SCOPE

This document describes the detailed design characteristics of the LACIE Phase II/III Status and Tracking System built for the PDP 11/45 and as modified for the CAMS Procedure 1. This System provides mechanisms to support the management of LACIE imagery processing and associated evaluation material. For each package of such material, the system maintains a record containing the history and the present status and location of the package as that package follows its track from one LACIE processing station to another. The System provides means for generating reports on status information and on statistical data about the flow of material through the LACIE stations.

This ASATS design is based upon achieving maximum use of the Regional Information Management System (RIMS) to perform ASATS functions. The functional design is described in terms of (1) data base design, (2) new RIMS software built to simplify implementation of ASATS, (3) software designed for ASATS which runs independantly of RIMS, and (4) a method for constructing standard ASATS reports.

1.2 BACKGROUND

A version of the ASATS was originally operated on COMSHARE's computer system. It was developed using the COMPOSIT'77 data management system. In order to reduce the cost of operating ASATS, it has now been implemented on the PDP 11/45.

In order that the transition from one computer system to another cause minimal impact, the standard update procedures (including update card formats) and standard reports were made as nearly as possible identical to the old ASATS system. Because a

different data management system was used for the PDP 11/45 than for the COMSHARE version, the ad hoc query and update processes are different. But the PDP 11/45 version provides similar capabilities for ad hoc query and update. Implementation of the PDP 11/45 version is based upon the same requirement document (LEC-8675) that was previously used for implementing the COMSHARE version of ASATS.

The original version of the PDP 11/45 LACIE Phase III ASATS was completed in March 1977. Since then, additional requirements for status and tracking of computer products for the CAMS Procedure 1 have been received and implemented. These additional data and programming requirements were delineated in the "Detailed Design Specification for the Automatic Status and Tracking System Modifications for LACIE Procedure 1" (TIRF 77-0020), LEC-10529, JSC-12885.

1.3 SYSTEM DESCRIPTION

ASATS operates on the PDP 11/45 using the Regional Information Management System (RIMS). It also uses the RSX-11D Version 6.01 operating system. ASATS System hardware requirements include:

- PDP 11/45 with a minimum of 64K of storage (32K for RSX and 32K for RIMS and ASATS)
- Disk storage (size requirement depends upon the number of data base records)
- TTY compatible terminal for interactive work
- Card reader for standard update and reports
- Printer
- Card punch (requirement is satisfied by off-line capability)
- Two magnetic tape units

ASATS software requirements include:

- RSX-11D version 6.01 operating system
- FORTRAN IV-Plus (F4P) compiler for software maintenance
- Regional Information Management System (RIMS)

The ASATS software is composed of (1) special processors built for ASATS to facilitate the auditing of ASATS data base updates, (2) RIMS commands augmented to support specific ASATS requirements and (3) command files (sequences of RIMS commands) which generate specified reports, (4) data base definitions describing the ASATS data base to RIMS, (5) format descriptions describing input files and report formats, and (6) command files which control RSX-11D system utilities.

2. APPLICABLE DOCUMENTS

The following documents are applicable:

- Large Area Crop Inventory Experiment (LACIE) Phase III Automatic Status and Tracking System Specifications, Revision A, dated March 1977 (document LEC-8675, JSC-11401)
- RIMS Design Document, February 1976 (LEC-9564)
- RIMS Maintenance Document, October 1976 (LEC-9566)
- RIMS User Document, dated April 1977 (LEC-9301, Rev. A)
- TIRF No. 76-0085
- ASATS Functional Design Document, November 1976 (LEC-9861, JSC-11835)
- Operator's Guide for ASATS, March 1977 (LEC-10401, JSC-12729)
- ASATS Users' Guide, March 1977 (LEC-10148, JSC-12535, Rev. A)
- Detailed Design Specification for the Automatic Status and Tracking System Modifications for LACIE Procedure 1 (LEC-10529)

3. INTEGRATED TOP-LEVEL DESIGN

3.1 GENERAL

An overall picture of the PDP 11/45 Status and Tracking System is shown in Figure 3-1. The arrows in the diagram indicate flow of information.

Figures 3-2 through 3-6 show the System in more detail and illustrate the data paths which satisfy the major requirements specified for the System as described in the following sections. These figures break the system into units by logical function. Section 4 will describe the way that separate parts of these logical systems are actually sequenced. For example, the data base save operation shown in Figure 3-5 will occur every night at the end of an update report cycle; the data base recovery shown in Figure 3-5 is a stand-alone operation that will never occur unless the data base is damaged.

3.1.1 STANDARD UPDATE PROCESSING

The processing paths of the standard batch mode daily update and report cycle are shown in Figure 3-2. Information flow through these paths is controlled by command files and a batch control-card file.

The Preprocessor is a set of operations specially designed for the ASATS update card formats. It produces some of the required audit listings of the input cards and rearranges the cards for the proper sequence of processing by RIMS. RIMS makes the required updates to the data base and produces a file of information concerning all attempted updates. A Postprocessor uses that information to produce the rest of the required reports on attempted updates and to punch the cards and print the labels that are required by certain updates.

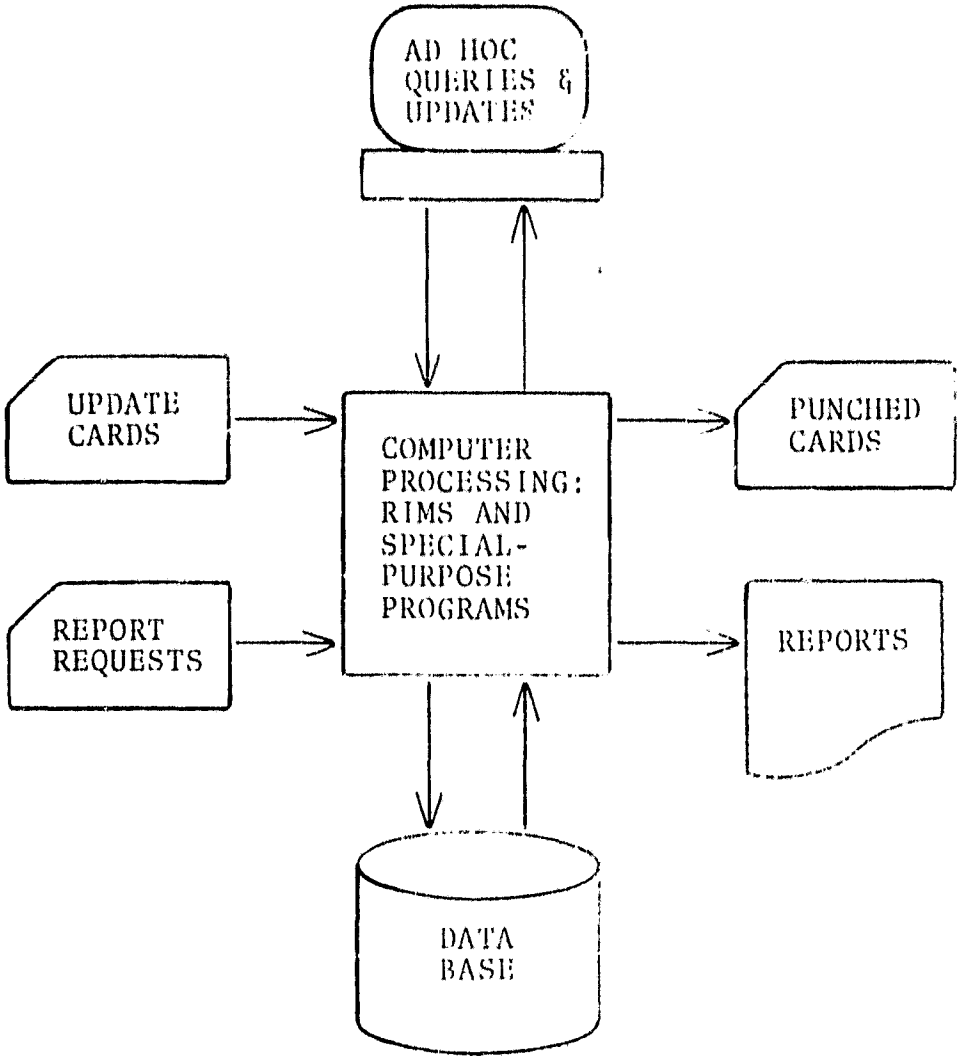


Figure 3-1.— Automatic Status and Tracking System — overall view.

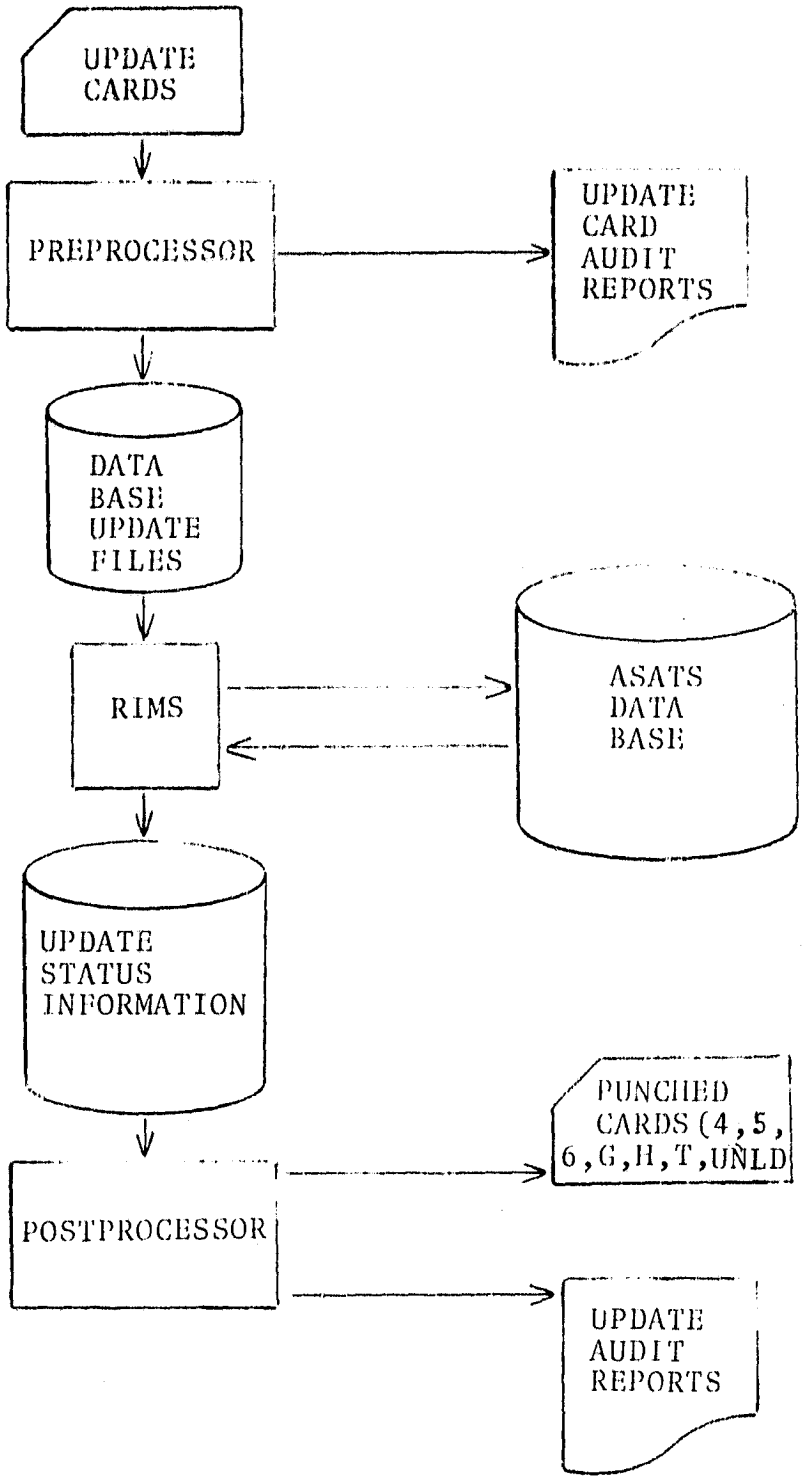


Figure 3-2.— Automatic Status and Tracking System — standard update processor subsystem.

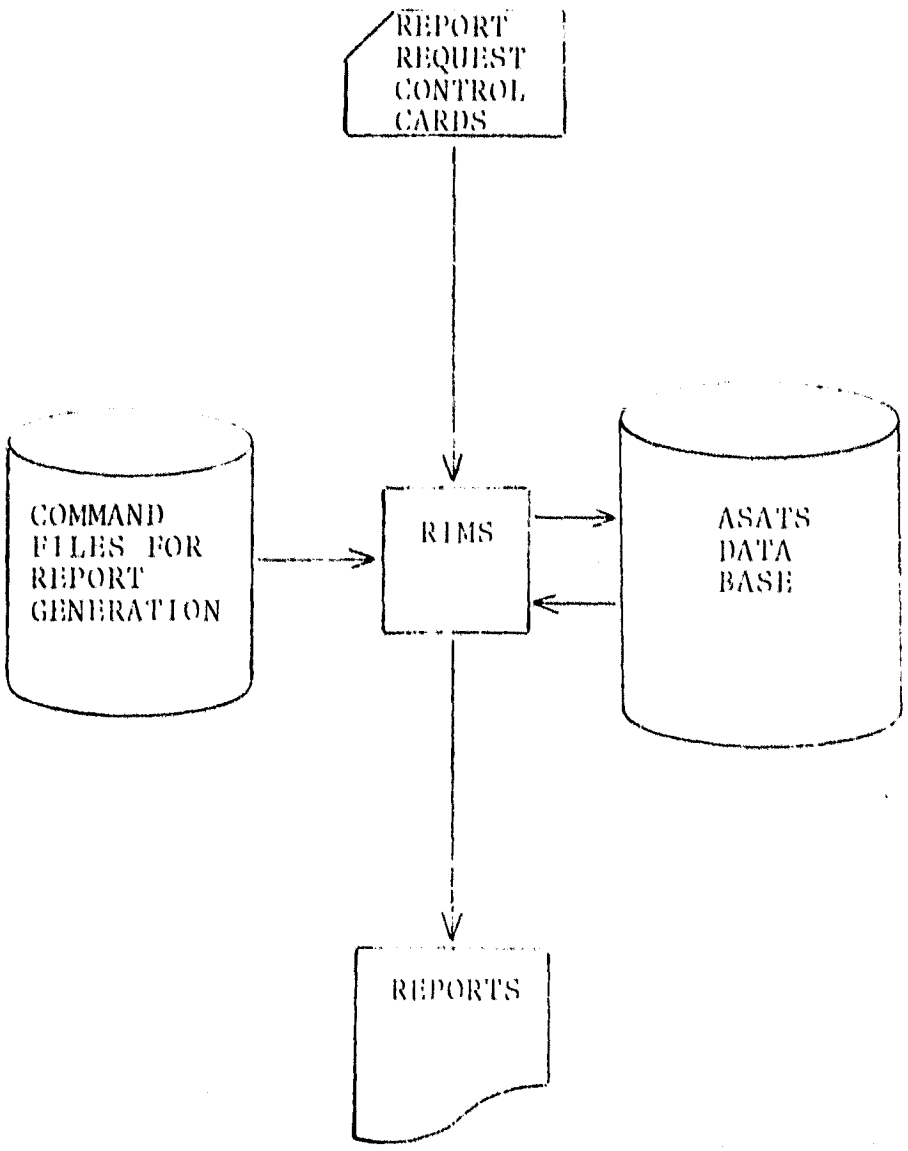


Figure 3-3.— Automatic Status and Tracking System — report generation subsystem.

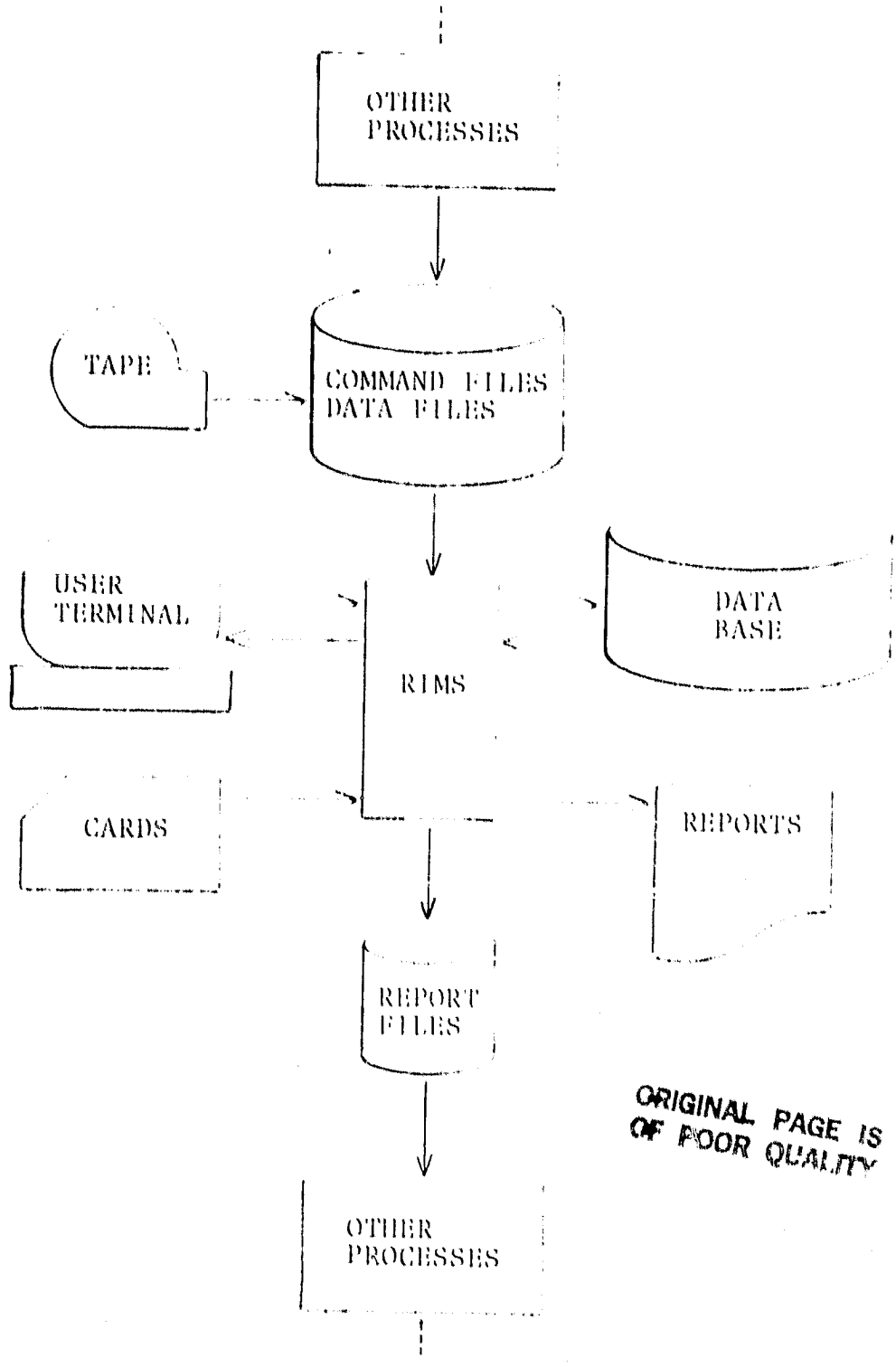


Figure 3-4.— Automatic Status and Tracking System — ad hoc query and report subsystem.

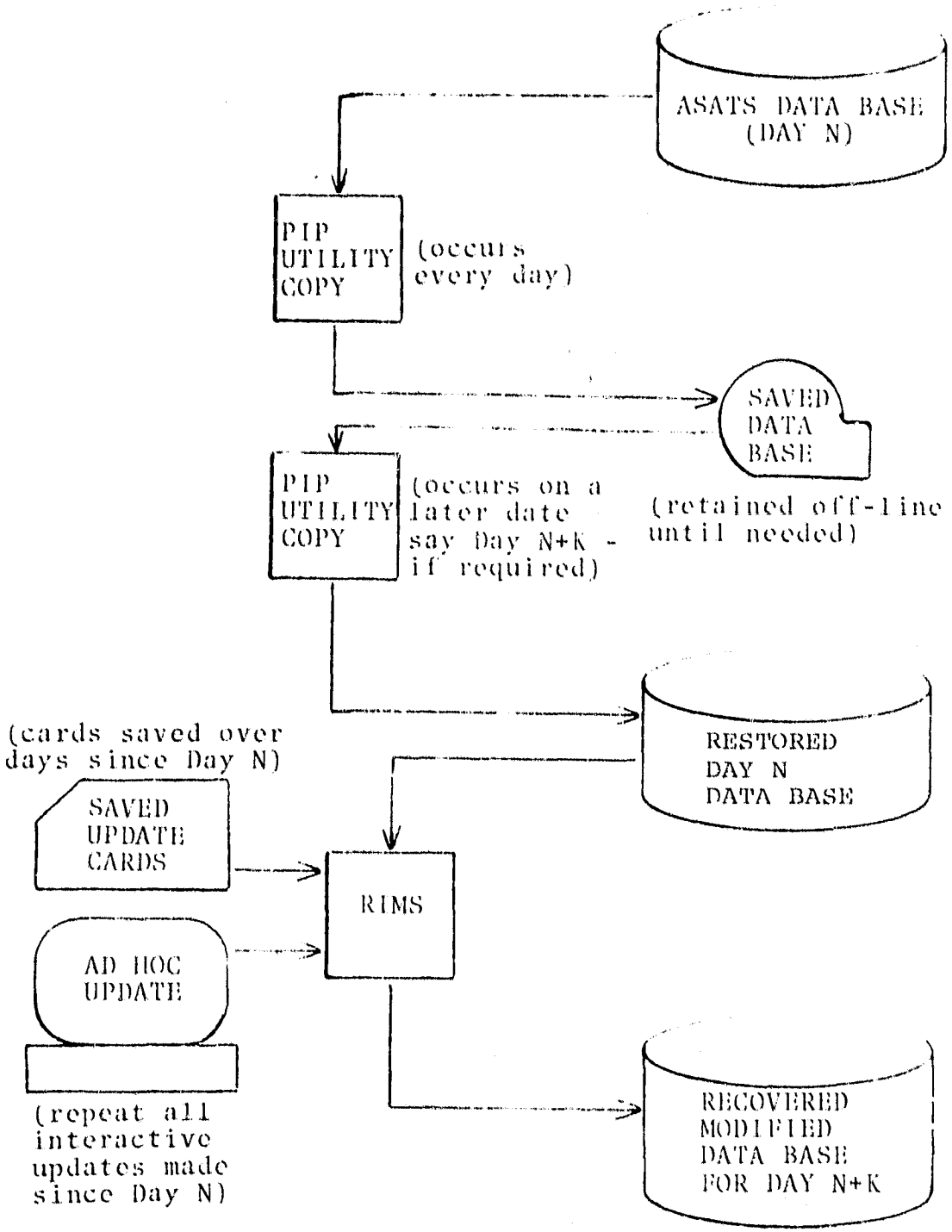


Figure 3-5.— Automatic Status and Tracking System — checkpoint — recovery process.

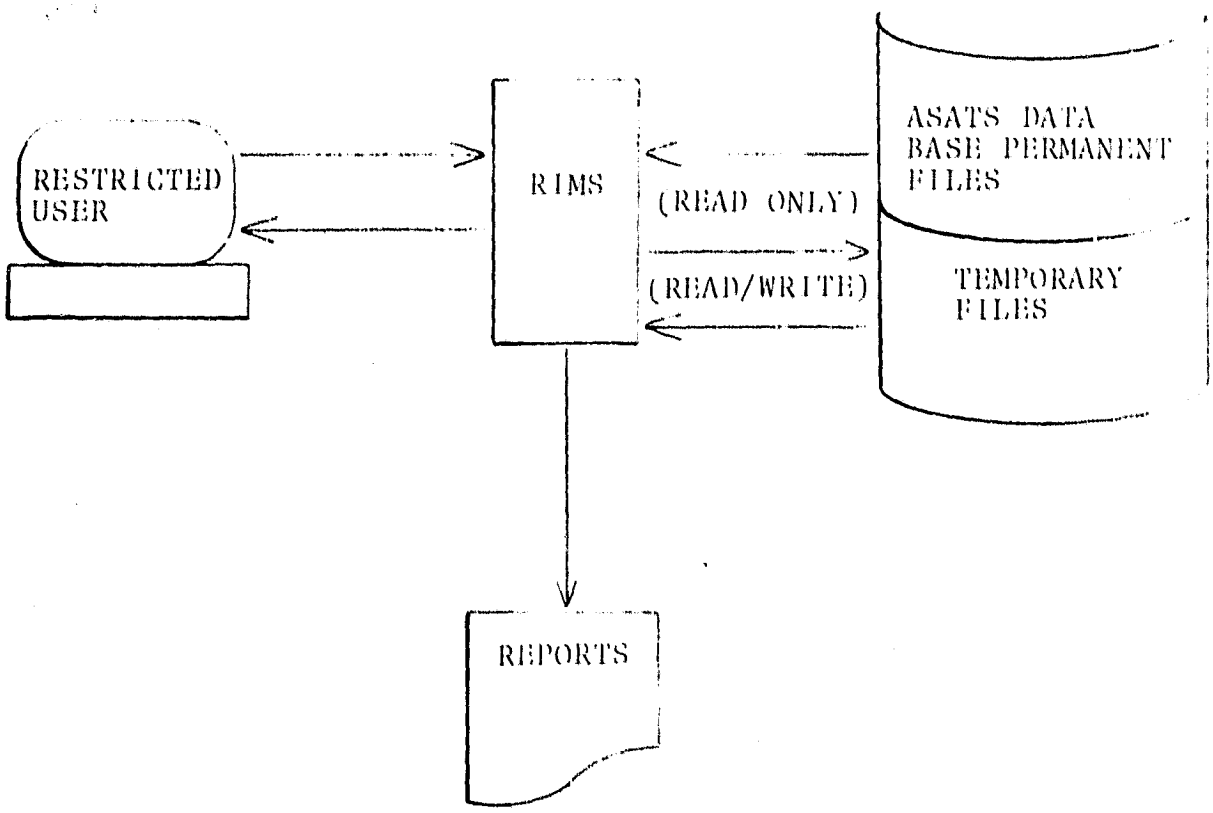


Figure 3-6.—Automatic Status and Tracking System — access control in ad-hoc report generation.

3.1.2 STANDARD REPORT GENERATION

Batch mode production of standard System reports is illustrated in Figure 3-3. For each of the standard reports there is a "canned" stream of RIMS commands to collect the information required for the report and put it into proper order. Selection of different reports as desired is made by different cards in the batch input control of RIMS.

3.1.3 AD HOC QUERY AND UPDATE

The generalized data management capability of RIMS can be used for several different purposes, as shown in Figure 3-4. Demand production of one-of-a-kind reports can be done through an interactive terminal or in batch mode controlled by cards. In either batch or interactive mode, RIMS can read a command stream or data file brought in from tape or produced by other processes. Information extracted by RIMS from the data base can be printed as a report or displayed at the interactive terminal or it can be used as input for other programs which might perform special analysis beyond the capabilities of RIMS itself. Some of the data flow paths shown in Figure 3-4 may be restricted for some users, as will be described in section 3.1.4.2.

3.1.4 DATA BASE INTEGRITY

3.1.4.1 Checkpoint - Restart Data Base Recovery

In spite of all economically feasible controls over hardware, software and procedures, the data base is vulnerable to damage or total destruction from such causes as computer system malfunction, flaws in physical storage media, or accidental running of improper updates. Figure 3-5 illustrates procedures which minimize the time and effort required to put the data base back into its proper current status after being damaged.

As shown, the data base is dumped to magnetic tape periodically (once per day, after making all of the day's updates is suggested; the time interval can be adjusted after gaining some experience with failure frequency for this system). These "checkpoint" dumps on tape are stored off-line, while the active data base is altered by batch and interactive updates on following days. If the active data base is damaged or lost, a checkpoint dumped tape (the most recent one dumped before the failure) is read back to on-line storage, which restores a previous day's active data base. That restored data base must then be modified by any batch-mode card updates and any interactive updates which have been made since the time the checkpoint dump was taken. Making those updates will bring the active data base to the condition it should have had if no failure had occurred.

3.1.4.2 Access Control

Accidental or malicious damage to the data base is minimized by controls which allow data base modifications to be made only by authorized personnel. Some users may produce reports without having permission to alter the data base. For these users, controls built into the RIMS software restrict data flow to the paths shown in Figure 3-6. As shown, the permanent files of the active data base can be read by a restricted user, but not written. Writing of any temporary files required to collect information for report production is allowed. A password system is used to identify users with unrestricted read-write access to the entire data base.

3.2 DATA BASE DESIGN

The data base for ASATS contains two primary types of data records. Each DAPTS record contains information about a segment that is pertinent to all acquisitions for that segment. The fields of the DAPTS records are described in Table 3-1. Each FLOCON record contains information about one particular acquisition

TABLE 3-1.–DAPTS RECORD FORMAT

Field Name	Description	Length (Char)	Start (Char)	End (Char)	Key	Source Card
SEG	Segment number	4	9	12		*
LPI	LACIE phase indicator	1	13	13		*
COUNTR	Country designator	6	14	19	X	*
REG	Region	2	22	23		*
ZONE	Zone	4	24	27		*
STR	Stratum	4	28	31		*
GD	Global designator	1	32	32		*
WV	Wheat variety	1	33	33	X	2
PC	Priority code	2	34	35	X	*
TY	Segment type	1	36	36		2
BIOW10	Biowindow 1 open (start date)	4	37	40		3
BIOW1C	Biowindow 1 close (end date)	4	41	44		3
BIOW20	Biowindow 2 open	4	45	48		3
BIOW2C	Biowindow 2 close	4	49	52		3
BIOW30	Biowindow 3 open	4	53	56		3
BIOW3C	Biowindow 3 close	4	57	60		3
BIOW40	Biowindow 4 open	4	61	64		3
BIOW4C	Biowindow 4 close	4	65	68		3
TOPO	Date topo map received	4	69	72		4
CROP	Date crop calendar received	4	73	76		5
ANCIL	Date ancillary data received	4	77	80		6
SSC	Segment status character	1	81	81	X	4,5,6
PROTYP	Process type	1	82	82	X	*
CDTAPE	CCIT & DTRM Tape Number	6	83	88		
TCARD	"T" card transaction date	4	89	92		T
LUP	Date of last change to this record	4	93	96		(last update)

and the processing of acquisition material packages by the LACIE work stations. The FLOCON records are described in Table 3-2. RIMS uses the data base to store information about the data base; i.e., format records (as described in RIMS documentation) for data records and for input and output records.

TABLE 3-2.— FLOCON RECORD FORMAT

Field Name	Description	Length (Char)	Start (Char)	End (Char)	Key	Source Card
SEG	Segment number	4	9	12		B
LPI	LACIE phase indicator	1	13	13		B
DATACQ	Acquisition date	4	14	17		B
BW	Biowindow	1	18	18	X	B
FF	Film flag	1	19	19		B
TAPE	GSFC tape number	6	20	25		B
GSFC	GSFC processing date	4	26	29		B
CANI	C&I update date	4	30	33		B
LPDLCO	Date film products received from LPDL	4	34	37		G
AICOMP	Date segment ready for CAMS pickup	4	38	41		H
PACKRE	Date packet received by CAMS	4	42	45		I
RUNSUB	Date FDB/batch data processing request submitted	4	46	49		J
RUNCT	Run count	1	50	50		(J)
PRODRE	Date batch products received by CAMS	4	51	54		K
REWORK	Date rework begun	4	55	58		M
RWKCT	Rework count	1	59	59		(M)
TOCAS	Date to CAS	4	60	63		X
CAMSBP	CAMS biophase	3	64	66		X
CATG	CAMS evaluation category	2	67	68	X	X
CURS1	Current film status	1	69	69	X	(last)
CURS2	Current product status	1	70	70	X	(last)
UTAPED	Date for unload tape	4	71	74		N
UTAPEN	Unload tape number	6	71	76		N
UNLOAD	Unload transaction date	4	77	80		N
LSD	Date of last change to this record	4	81	99	X	(last update)

3.3 RIMS MODIFICATIONS

This section identifies modifications and additions to RIMS which have been implemented to support ASATS. These changes are catagorized as follows:

a. Special Update Processor - This processor was defined because of the requirement to implement ASATS on the PDP 11/45 with no changes of input from the format for ASATS on COMSHARE using Composit '77. The processor handles all standard ASATS input cards.

b. Relational Retrieval to Eliminate Redundant Data - The data base for ASATS has a hierarchical relationship between DAPTS records and FLOCON records. RIMS commands have been implemented to create a set of related FLOCON records for given DAPTS records, and to create a set of related DAPTS records for given FLOCON records. Also, a command for displaying records containing data from both a FLOCON record and the related DAPTS record has been provided.

c. Access Control - A command has been provided which specifies which RIMS commands are allowed for each of the access control words assigned to different users. Also, the system includes a modification that requires a user to identify himself with an access control word at the beginning of a session.

d. Computation on Data Base Content - A new command allows the user to sum fields, field differences, compute a mean, and compute a standard deviation for date fields in a set of records.

e. Header Control for Reports - Commands have been provided to allow the user to specify report headers and to provide textual description for the number of entries in a set.

3.3.1 ASATS STANDARD DATA BASE UPDATES

This section describes the ASATS data base update processor. It also includes a description of an update command built specifically for ASATS and describes the use of input formats to specify processing for individual record types.

3.3.1.1 Special Update Processor

The Special Update Processor is a stand-alone program. It processes all standard ASATS updates. It accepts the following commands:

- BE - Begin
- RF - Reassign File
- UP - Special ASATS update command
- EN - End
- RE - Reads processing description for ASATS cards

The construction of this Processor uses all standard RIMS commands except the main program and subroutine update (which processes the UP command). The construction of this processor as a task separate from RIMS allows better core utilization, hence better system performance. Before executing any UP command, an RE command must be executed to read the process description cards which describe the processing for each status and tracking card type.

3.3.1.2 Special Update Command

Purpose: Updates data base from a set of input cards. Specific update operations are a function of card type (specified in the second character of each card), data base format, and the input format. The input format and the data base to be used are a function of the card type and LACIE phase.

Input:

- Commands: Processing is begun by a UP command
- Status and tracking input cards: Any of the current 21 types of ASATS update cards (except Q card) are processed sequentially until an EOF.
- EOF: Processing of an input file is ended by an end-of-file or a blank record on the input file

Output: Besides updating the ASATS data base, the following information is recorded sequentially on a file:

- Rejected input cards
  - Cards for which the required DAPTS record does not exist (for *, 2, 3, 4, 5, 6, B, N and T cards)
  - Cards for which the required FLOCON record does not exist
  - Cards for which the FLOCON record has not reached required state for particular type of update
- Accepted input cards which created new DAPTS records
- Punched card images

Processing: The required processing is a function of card type. Card types are categorized as follows:

- Category 1 - card types *, 2, 3 (in sets)
- Category 2 - card types *, 2, 4, 5 and 6
- Category 3 - card type 3
- Category 4 - card type B
- Category 5 - other card types
- Category 6 - card types N and T

There is a generalized function for adding new records and updating existing records. This function, which is driven by input formats, data base formats, and card type, adds or modifies the specified record. The general steps of processing input cards are:

- Read input card
- Generate record ID
- Generate external (input) format ID from table
- Retrieve record
- Retrieve formats
- Either add or update the record
- Output card image reflecting success or error

Figure 3-7 depicts the flow of this process and variations dependent upon category of card type.

Table 3-3 indicates the data used for generating a record depending on input category. The input format is a function of the card type. Table 3-4 indicates the action to be taken upon a record retrieval failure.

The input processing for all fields of each card type is as defined in the ASATS requirements document except for Segment Status Character in DAPTS records and Acquisition Status character in FLOCON records. The information from Table 3-3 is used to set the segment status character. The Acquisition Status Character is set to the card type. These fields are used for generating the current station and status. Their use is described in paragraph 3.6

The type of operation, an add or a modify, to be performed is a function of record type and whether or not a record already

exists. The input format for the card type identifies the fields it updates and the field's data type. Table 3-5 describes the processing for field types on input.

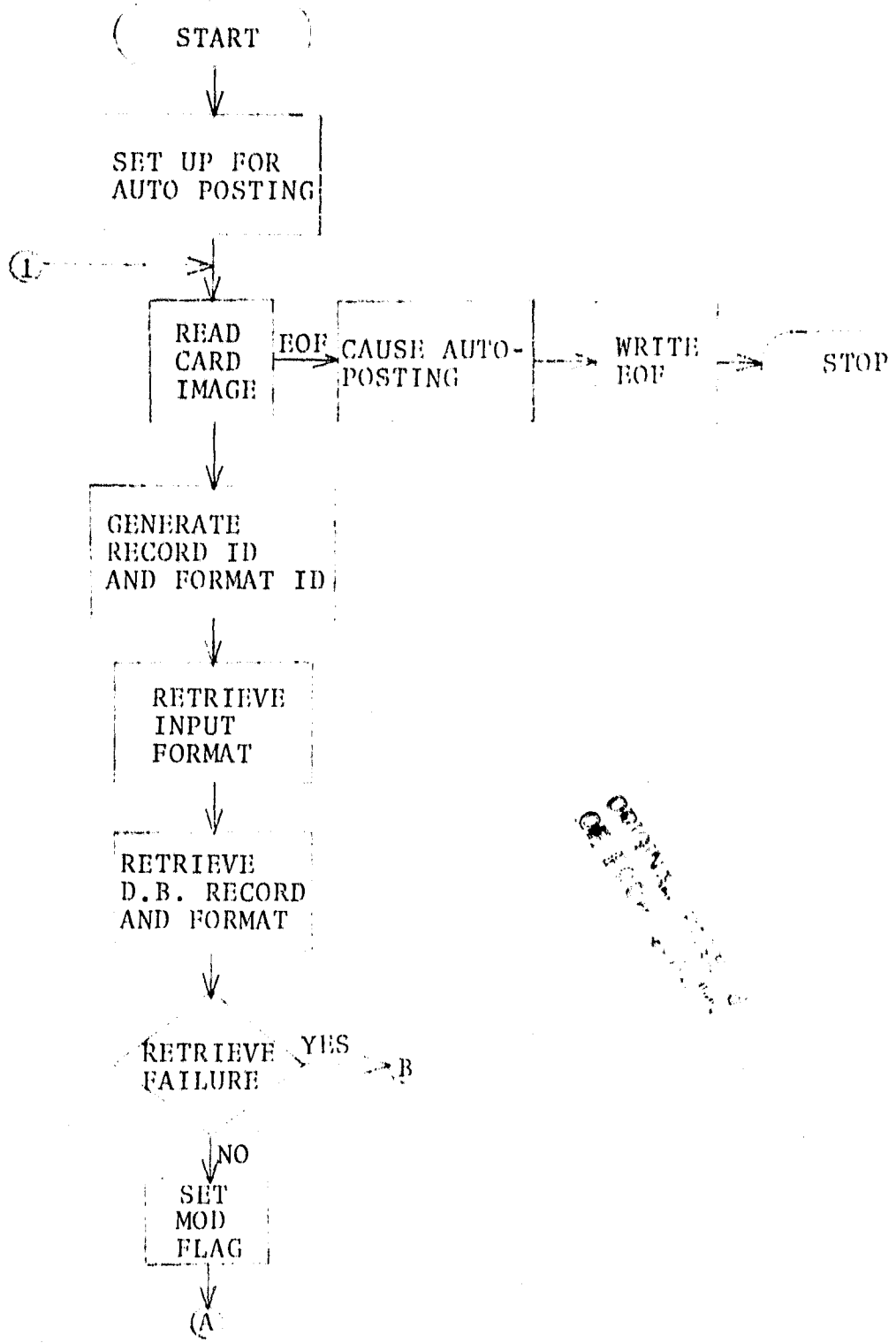


Figure 3-7.— Special Update Processor Flow.

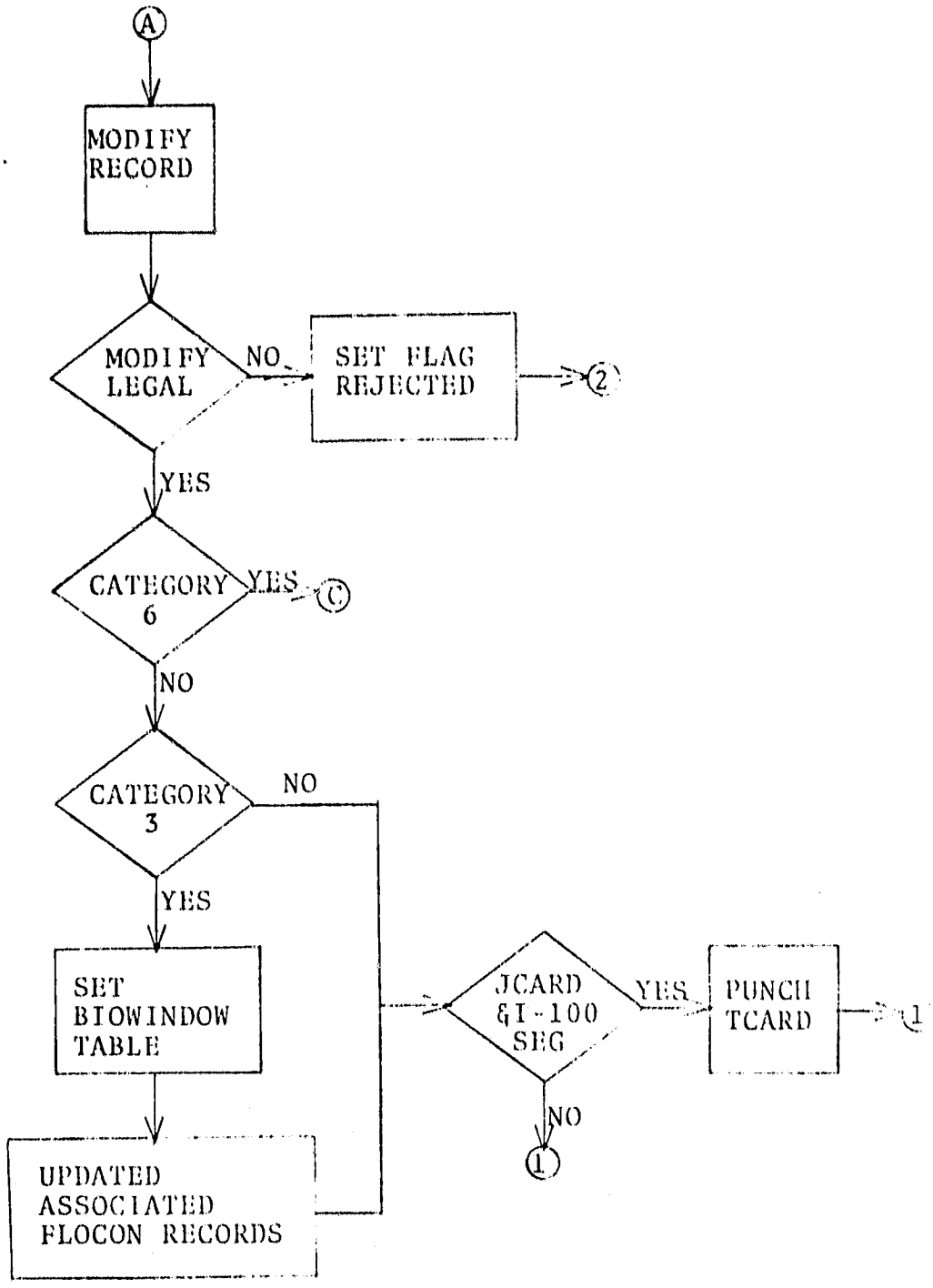


Figure 3-7 .— Continued.

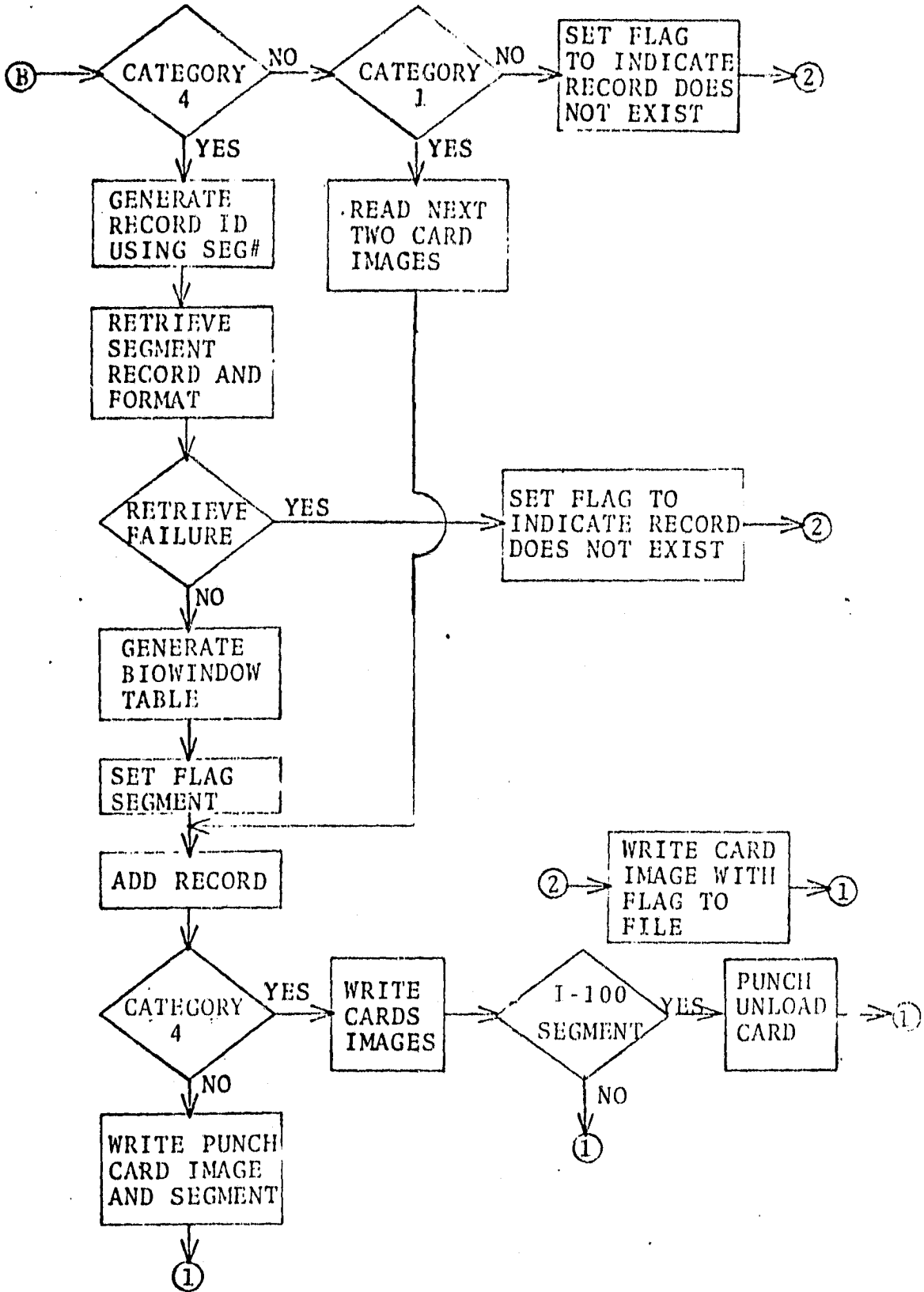


Figure 3-7.— Continued

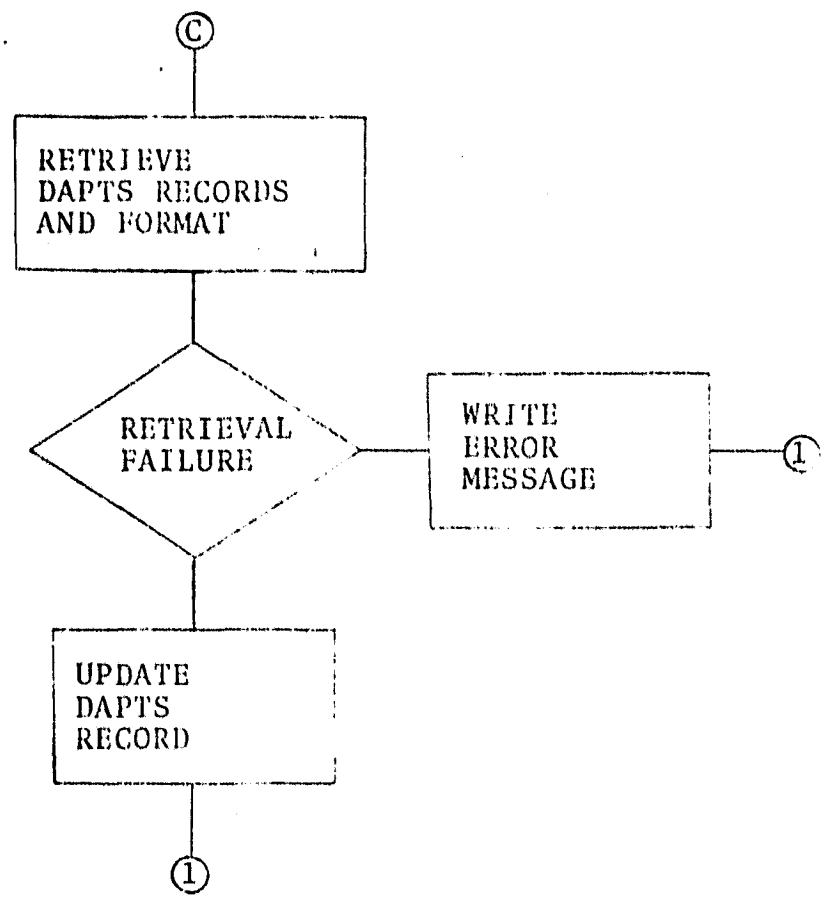


Figure 3-7.-- Concluded.

TABLE 3-3.-INPUT TRANSLATION TABLE FOR GENERATING SEGMENT STATUS CHARACTER

Existing value	Card type 4	Card type 5	Card type 6
0	1	2	4
1	1	3	5
2	3	2	6
3	3	3	7
4	5	6	4
5	5	7	5
6	7	6	6
7	7	7	7

TABLE 3-4.-SPECIAL UPDATE DESCRIPTION

Category	Method of Identifying Record	Action on Retrieval Failure	Generate Biowindow Table	Type of Operation	Additional Processing
1	SEG #	Success: Update (1b) Fail: Add		Add	
1b				Update	
2	SEG #	Fail: Generate Error	Yes	Update	Set Bio-window Field in Acq. Records
3	SEG #	Fail: Generate Error		Update	
4	SEG # & Acq. Date	Fail: 4B Success: Update		Update	
4b	SEG #	Fail: Generate Error	Yes	Add (Acq. Record)	
5	SEG # & Acq. Date	Fail: Generate Error		Update Update	
6	SEG # & Acq. Date	Fail: Generate Error		Update (Both DAPTS & FLOCON	

TABLE 3-5.-USE OF DATA TYPES ON INPUT

Data Type	Processing
0	Alpha - update associated data base field as alpha
1	Integer - update associated data base field as integer (standard RIMS data type, but not used for ASATS)
3	Set data base field value from status table according to card type and existing value
4	Alpha but don't update data base when input is blank
5	Record I.D. field (no action)
6	Increment data base field value by 1 on input (Integer field)
7	Reject input if associated data base field is blank
8	Set data base value as a function of biowindow table and acquisition date

If an error condition occurs when processing an input card, an error file unit number is put in column 2 when the image is written to the message file. The input card images for new DAPTS records are also written to the message file; in them, column 2 is the unit number for the new DAPTS record report file.

Additional processing required by category 3 is the selection of FLOCON records of the same segment and the updating of their biowindow fields.

3.3.1.3 Use of Input Formats

An input format is required to describe certain processing associated with each record type. This description includes:

- Identification of each field affecting the update process
- Type identification of each field, which defines how the field affects the update process (see section 3.3.1.1).
- Starting location and length of each field of information on the data card (it should be noted that some fields affect the processing, but do not exist on the data card, therefore do not have a starting location and length).

The field names used must correspond to the field names used in the data base definition (see Tables 3-1 and 3-2). Field location and field use on input cards are described in the ASATS requirements documents and in the ASATS User's Guide (Rev. A).

Implementation using input format definitions allows for simple accommodation of most types of changes in input requirements. The addition of a new card type would require a modification of tables within the special update command processor. The B and 3 type input cards do have codes which are unique to them.

3.3.2 SPECIAL COMMANDS FOR ANNOTATING REPORTS

A new command (header) allows the user to include text which is not a part of the data base and to include printer carriage controls. Another new command is included which allows the user to print a set count with text which he specifies.

3.3.2.1 Header Command

Purpose: To insert header or comment on report.

Input: HDN, Text

where: HD is the command mnenomic
N is the number of input lines (1 or 2)
Text is the header contents

Output: N lines of text

Processing: Text is transferred from the command file to the report file with the first character after the comma being used for carriage control. Standard FORTRAN conventions are used for carriage control.

3.3.2.2 Count Command

Purpose: To print the number of entries in a set along with text comment.

Input: SCSN,LOC, Text

where: SC is the command
SN is the set number for the count which is to be printed
LOC is the column count where the set count is to be printed
Text is the text data to be printed

Processing: Text is transferred from the command file to the report file and the number of entries is printed with it. The first text character is used for carriage control.

3.3.3 SOFTWARE TO ELIMINATE REDUNDANT STORAGE

The following new RIMS commands were designed in order that certain data does not have to be stored redundantly in both DAPTS and FLOCON records. The DAPTS and FLOCON records of the ASATS data base have a hierarchical relationship. The DAPTS record (the parent record) contains information common to possible several FLOCON records.

3.3.3.1 Generate Parent Set

Purpose: To generate a set of parent records (DAPTS records, for ASATS) from a child set (FLOCON records, for ASATS)

Input: GPSN

where: GP is the command mnemonic
SN is a temporary child set number

Output: A set (next available temporary set number) of parent records

Processing: The parent record ID for each child record ID in the input set is placed in the output set.

3.3.3.2 Generate Children Set

Purpose: To generate a children set (FLOCON records in ASATS) for a set of parent records (DAPTS record in ASATS).

Input: GCSN

where: GC is the command mnemonic
SN is the temporary parent set number

Output: Children set (whose set number is the next available temporary set number)

Processing: The record pointer address is computed for each record ID in the parent set. The addresses for potential children records (up to 16 records following the parent records) is searched to discover whether children records exist. Each child ID that is found is then put into the output set.

3.3.3.3 Display Data from Two Data Base Levels

Purpose: To display a set of FLOCON records with selected information from the associated DAPTS records, as specified by a format (Joint Display Formatted).

Input: JFSN, FMT

where: JF is the command mnemonic
SN is the temporary set number for a set of FLOCON records
FMT is the format ID for displaying data

Output: Specified set in specified format on report file.

Processing: The format for displaying data is retrieved. Then for each record ID in the input set, the following actions are taken:

- Get record ID from set
- Retrieve record and its format
- Transfer data from the child (FLOCON) record to an output buffer according to the display format
- Generate record ID for associated parent (DAPTS) record
- Retrieve DAPTS record and associated format
- Transfer data from the DAPTS record to output buffer according to the display format
- Write output buffer to the report file

3.3.4 ACCESS CONTROL

The following new commands, software, and RIMS modifications are defined to satisfy access control requirements. Each user will have an access control word. It will control which RIMS commands he can or cannot use.

3.3.4.1 Add Access Control Word

Purpose: To add an access control password for the data base.

Input: S+
PASSWORD>XXXXX
BIT MASK>1110111....

where: S+ is the command mnemonic
XXXXX is an 8-character access control word
1110111... is a 24-character string identifying which commands the user having this access code can utilize.

Output: Access code and character string are stored in data base.

Processing: A record is generated containing the character string. A key is generated by hashing the character string. The record is posted to that key. NOTE: The key cannot be reconstructed by the display from the expand function.

3.3.4.2 Delete Access Control Word

Purpose: To delete an access control password.

Input: S-
PASSWORD> XXXXX

where: S- is the command mnemonic
XXXXX is the access control word

Output: No reply - access code and character string are deleted from the data base.

Processing: The record posted to the password is deleted and the key associated with the control word is deleted.

3.3.4.3 Modifications to Allow Password Control

Purpose: To allow identification of a user to RIMS by access control word following first begin command.

Output: PASSWORD> (prompt to user)

Input: Access Control password

Processing: The record associated with the control word is retrieved. If the given control password does not exist, RIMS is aborted.

3.3.4.4 Modification to RIMS Control Routine for Access Control

Purpose: To prevent execution of unauthorized commands.

Output: Message file - 'illegal command'.

Processing: Before calling the subroutine to process the associated command, the character string containing indicators of user's authorized commands is examined. If the user is not authorized to use the command, the message is output and the subroutine is not executed.

3.3.5 ARITHMETIC OPERATIONS

A new RIMS command has been included to provide the typical statistical and arithmetic operations required by ASATS in the ad hoc environment.

3.3.5.1 Compute Command

Purpose: to provide for a set of records (1) summation of identified field, (2) mean of the difference of two fields, (3) standard deviation of the differences of two fields.

Input: ARSN, RID, FN_1, FN_2

where: AR is the command mnemonic
SN is the set number for which all computations are to be performed
RID is the record ID for the record where the results of the computation are to be stored
FN_1 is a field name
FN_2 is a field name

Output: Sum of both fields, mean of difference, standard deviation of difference, count of the number of records used in mean computation, and the count of the total number of records are stored in the specified record.

Processing: Each record in a set is retrieved and processed.

The following computation is performed for each record.

- Each field is summed over all records in which both fields are not blank.
- A count of the number of records is maintained.
- For each record where either field is blank
  - Count of the records is maintained
  - Sum of the difference of the two fields is maintained

After all records have been processed, the mean and the standard deviation of the differences are computed. All computed results are then stored in the specified record.

3.3.6 MODIFICATION TO EXISTING COMMANDS

In order to implement standard reports by executing a RIMS command file containing the RIMS commands for reports, two new capabilities have been implemented. They are:

- The ability to generate null sets
- The ability to assign a file by name

The following modifications provide these capabilities.

3.3.6.1 Modification to Reassign File

Input modifications: The previous syntax was RFIU,EU. The syntax has been modified to allow another form: RFIU,EU,FNA.

where: RF is the command
IU is the internal unit number
EU is the external unit number
FNA is the file name

so that if IU is 0 (zero), then EU is set to the included file name.

Output modifications: None

Processing modification: If the internal unit is zero, the external unit is closed then re-opened with the specified file name. Otherwise, processing is the same.

3.3.6.2 Null Set Option Addition

Input: ZZOP

where ZZ is the command mnemonic
OP is the option: 1 for null sets to be kept; zero for null sets not kept. At sign-on, the default mode of the system will be for null sets not to be kept.

Output: None

Processing: A flag is set in a common block to indicate the null set mode.

3.4 THE PREPROCESSOR

3.4.1 PURPOSE

The Preprocessor produces several of the required audit report listings of the input update cards and acts as an interface between the unsorted, unedited input form and the form of input required for the RIMS data management software.

3.4.2 INPUT

The Preprocessor accepts update cards of all types as described in the ASATS Specifications Document, LEC-8675 (Rev. A), and ASATS LACIE Procedure 1 Detailed Design Specification, LEC-10529. The cards may be in any order and must include one and only one Q card. The input update cards for the Preprocessor make up the last of three files read in by the operator before each daily update and report batch run. (The first two files are for report (generation.)

3.4.3 OUTPUT

The Preprocessor outputs two types of data. The first type is the listing of input cards:

- A listing of all cards in the order of their input
- A listing of all cards having invalid card types
- A listing of all cards rejected as duplicates
- A listing of all cards submitted for this update run, sorted into order by card type.

The second type of output from the Preprocessor is a set of files containing update card images which are combined to drive the RIMS system for actual update of the data base. The files are:

- A file of all sets of *, 2, 3 cards, where a complete set is given for any segment
- A file of *, 2, 3 cards in which only one or two of the cards appears for any segment
- A file of other update cards sorted into the order: 4, 5, 6, B, 7, 8, 9, G, H, I, N, J, K, M, T, X, U.

3.4.4 DESCRIPTION OF PROCESS

The Preprocessor consists of a sequence of operations performed by special-purpose FORTRAN programs and by calls to the RSX-11D system SORT utility program. Data flow for this sequence of operations is shown in Figure 3-8.

The first operation shown, gives two of the input listings and separates update cards by LACIE phase into separate files. In this operation, cards with invalid type or LACIE phase are rejected, and a check is made to be sure there is one Q card.

The second operation is performed by the RSX-11D system SORT utility, to put the cards into order by card type.

The next operation shown is a separation of cards by type. The type *, 2, 3 cards will be sorted by segment number to find those segments for which all three are present and which may, therefore, be new segments for entry among the DAPTS records. Cards not part of a set will be sorted back into card type order with * cards first, then 2 cards, then the 3 cards. Cards of other types (4, 5, 6, B, 7, ···) are put into another separate file.

3.5 THE POSTPROCESSOR

3.5.1 PURPOSE

The Postprocessor is required because of core space limitations on the number of output files that can be defined in RIMS. The RIMS update task for ASATS writes several logical output files into one actual output file and attaches a flag to each record to allow the Postprocessor to separate the output files.

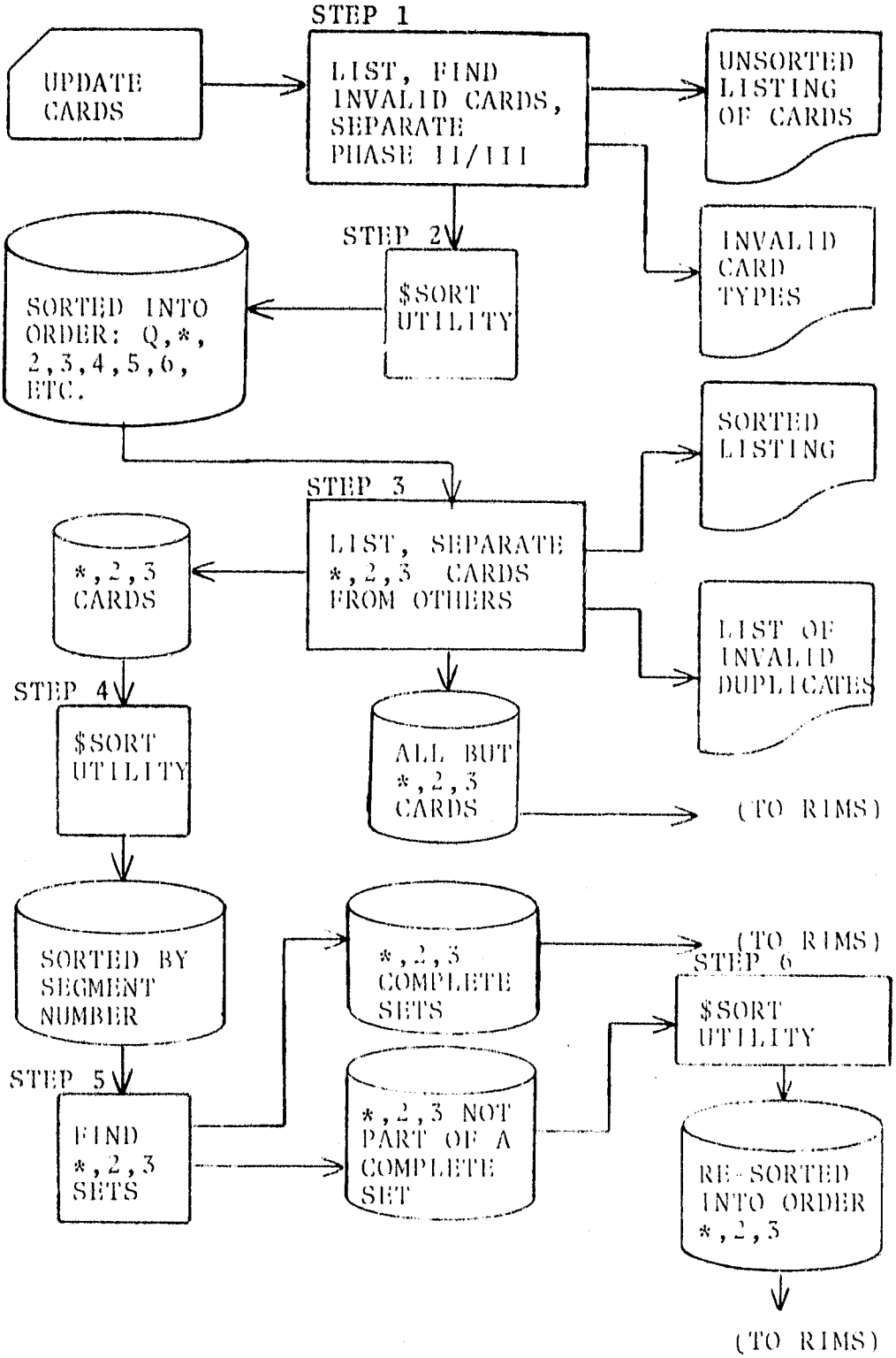


FIGURE 3-8

THE PREPROCESSOR

3.5.2 INPUT

Input to the Postprocessor is a file of information about updates made or attempted by RIMS.

3.5.3 OUTPUT

Output from the Postprocessor consists of one file of card images to be punched and several files of report listings, as follows:

- Cards punched (4, 5, 6 cards for each *, 2, 3 set that created a new DAPTS segment record; G, H, and UNLD cards punched for each B card that created a new acquisition (FLOCON-record); and and T card punched from each I-100 segment J card submitted.
- A listing of the cards punched
- A listing of all new DAPTS segment records added
- A listing of invalid acquisitions (B cards for which no DAPTS record has the same segment number)
- A listing of invalid attempts to modify a DAPTS record (modifications from *, 2, 3, 4, 5, 6 cards such that no existing DAPTS record has the given segment number)
- A listing of invalid modifications to FLOCON records (no matching segment number and/or acquisition date for input G, H, I, J, K, M, N, T, U, X, 7, 8, or 9 cards) or a required data base field has not previously been set (as required for card types H, I, J, etc.).
- Packet labels

3.5.4 DESCRIPTION OF PROCESS

The Postprocessor will read the output file from RIMS and write into the proper output file(s) chosen on the basis of a flag included with each record from RIMS. See figure 3-9.

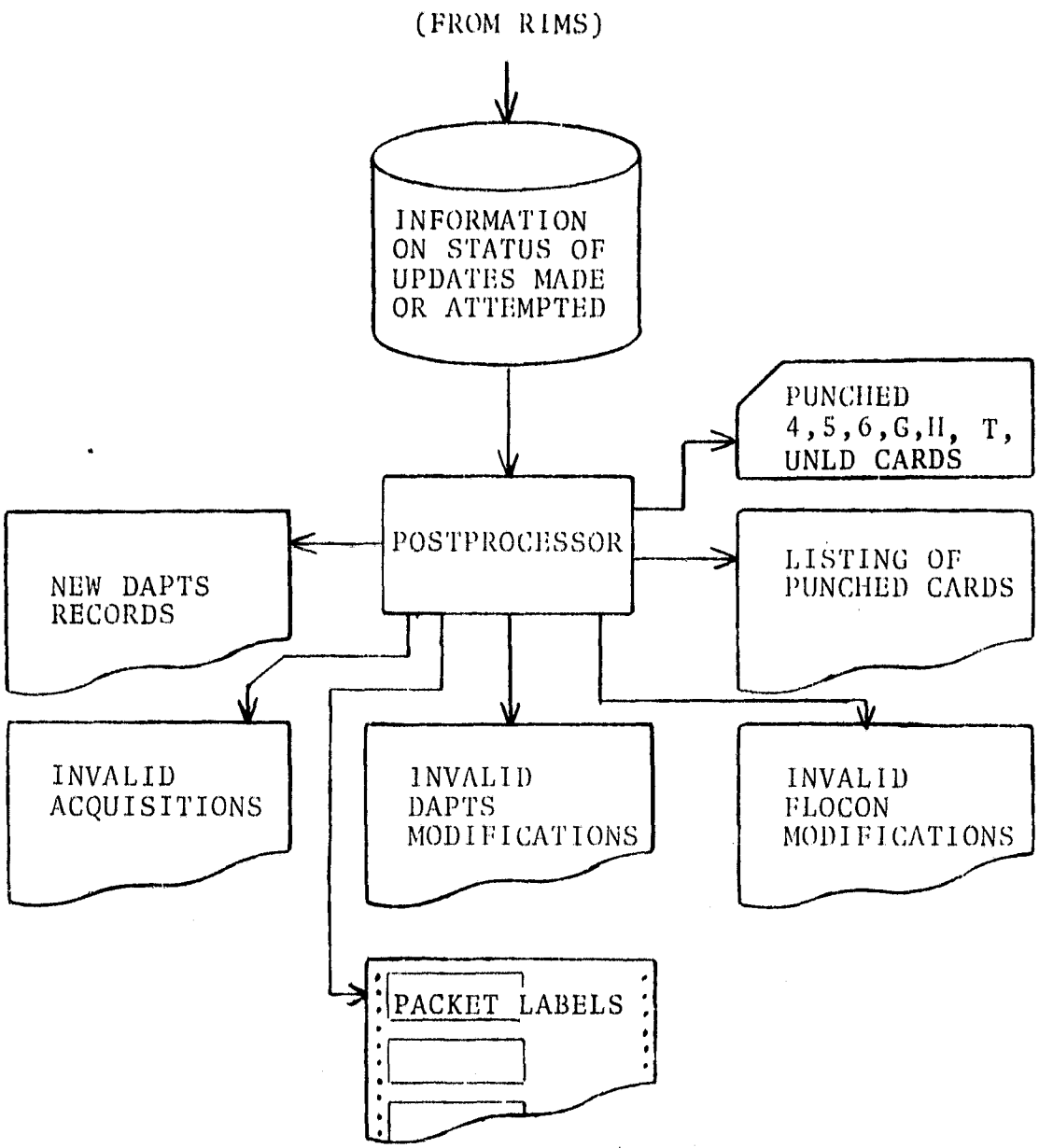


FIGURE 3-9

ASATS - THE POSTPROCESSOR

3.6 ASATS STANDARD REPORTS

All standard reports identified, as required for ASATS, can be produced using RIMS commands (as augmented in section 3.4 of this document) and the added output translation function. Once a data base has been established as described in this document, RIMS provides the functional capabilities required for these reports.

In order to satisfy the standard report and ad hoc report requirements using the data base design as described in section 3.2, it is necessary to perform output translation for some data elements. This translation generates the output field value as a function of the input field value using a predefined table.

Three tables are required for the necessary output translations. They are:

- Film products status table as a function of Acquisition Status Character CURS1.
- Computer products status table as a function of Acquisition Status Character CURS2.
- Table indicating crop calendar, ancillary data, and topographic data status as a function of the segment status character.

Two additional data types are required for output processing. They are:

- Data type 5 - Perform output translation using film products status table.
- Data type 4 - Perform output translation using computed products status table.

For each data type, the input format specifies the location of

data to be used for the table look-up. The output format specifies the location where the table results are to be placed in the output record.

These records have been implemented by building separate files containing the necessary RIMS commands for each report. The report can then be produced by assigning the file as a RIMS command file. The reports are initiated as part of the batch run by a "starter" file that is copied to the standard RIMS command file. The starter file reassigns the command file to the actual report file.

The requirements for each report have been translated directly into RIMS commands. Individual report command files are described in section 4.

4. CONTROL FILES

This section describes the files which control the operational ASATS system: utility command files, batch run control card files, and "canned" RIMS command files. Command files used to build tasks will be described in section 6, and data files are described in section 5.

4.1 PIP UTILITY COMMAND FILES

4.1.1 UP.CMD

This file of PIP commands uses three files from the daily input card deck to prepare files for the daily update and report run: the date to be on reports, the reports desired, and the updates to be made to the data base. The text of this file appears in figure 4-1.

This command file is invoked by the operator:

```
MCR>HEL [(ASATS UIC)]
MCR>PIP @UP
```

The card deck to be processed must contain three files (each terminated by the standard PDP-11 RSX-11D end-of-file card);

1. A RIMS HD command for the header date line in reports
2. A file of PIP commands that copy RIMS report command files into successive versions of the file REP.COM
3. The standard ASATS update cards, including one Q-type card.

The output files produced are

1. File DATE.COM which contains the HD command giving the date and an RF13,13 command.
2. As many versions of REP.COM as were specified by the input, plus five additional dummy versions to prevent RIMS from trying to read a nonexistent file.

```
DATE.COM=ENDFILE
DATE.COM;*/DE
DATE.COM=CR;
DATE.COM=RF1313.COM/AP
BLD.CMD=ENDFILE
BLD.CMD;*/DE
BLD.CMD=ENDFILE
BLD.CMD=CR;
REP.COM=ENDFILE
REP.COM;*/DE
@BLD.CMD
REP.COM=ENDFILE
REP.COM=ENDFILE
REP.COM=ENDFILE
REP.COM=ENDFILE
REP.COM=ENDFILE
PPF11.CRE=ENDFILE
PPF11.CRE;*/DE
PPF11.CRE=CR;
```

Figure 4-1.—The Update Card Read Command File, UP.CMD.

3. The file PPF11.CRE of ASATS update card images.

In the process of creating the above files, any files left over from previous days are deleted. Deletion of files is always preceded by creation of a dummy file (a copy of the empty file ENDFILE.;1) so that the delete operation will not give any meaningless error messages to the operator.

4.1.2 LA.CMD

This file is used by the operator to print packet labels on the line printer. The text of this file appears in Figure 4-2. It is invoked with the command:

MCR>PIP @LA

The input is the label file created by the Postprocessor, POST8.ZIP. It is copied to the line printer.

4.1.3 SAMTn.CMD

There are two files which can be used to do the daily save of the data base. The file SAMTØ.CMD saves it on MTØ: and SAMT1.CMD saves it on MT1:. The file SAMTØ.CMD is shown in Figure 4-3.

The operator must initialize and mount a tape, and then issue a command:

MCR>PIP @SAMTn

The command file copies the DATE.COM file to the tape as a check of the date the tape was made, and then copies all the data base files to the tape. The command file then gets a directory of the tape and saves that directory on disk, prints it on the line printer and prints it on the terminal, to assure the operator that the files have been saved on tape.

```
LPI#POSTA.ZIP
```

Figure 4-2.—The Packet Label Print Command File, LA.CMD.

```
MT0:=DATE.*/*
MT0:=XX.*/*
LP.TES=MT0:*.*/*/LI
LP:=LP.TES
TI:=LP.TFS
```

Figure 4-3.—The Data Base Save Command File, SAMTØ.CMD.

4.2 ASATS.BIS, THE BATCH RUN CONTROL CARD FILE

The control card file for the daily batch run of updates and reports is designed to satisfy several criteria which prevent it from always being straightforward, namely:

- It should not give the operator any specious error messages- e.g., if a file which may or may not be left over from a previous run must be deleted, the message that the file does not exist should be suppressed.
- The operator should interact with the operations as little as possible, and his choice of interaction should always be clear.
- Status information should be given periodically to allow restart, if possible.
- The many files created during a run should be deleted automatically, within the run or on the next run.

Keeping those principles in mind to explain the other "extra" operations, the basic required sequence of operations is this:

- Run step 1 of the preprocessor
- Sort the separate LACIE Phase update card files into order by card type
- Run step 3 of the preprocessor
- Sort the *,2,3 card files into sets by segment number
- Run step 5 of the preprocessor to recover complete sets of *,2,3 cards
- Put the non-set *,2,3 cards back sorted by card type order
- Append the separate update files into a single file for each LACIE Phase
- Run the ASATS task (RIMS data base update) for LACIE Phase 3. (Phase 2 updates have not yet been incorporated as an automatic operation in the daily run)

- Run the postprocessor to convert the single report file from the update run into 8 separate files; 6 audit listings, the packet labels, and a file of card images to be punched.
- Run RIMS five times as a batch mode task to produce as many as five reports requested for this run. The input data for the batch run is all from disk files. The data base is permanently resident on disk and other data input (the date to be put onto reports, the RIMS command files that start reports, and the update cards) are read in immediately prior to the batch run (see the description of the PIP command file UP.CMD). Much of the output of the batch run goes directly to line printer spool files. Three of the outputs are left in disk files and copied to output media by the operator: packet labels are copied to label stock in the line printer with PIP command file LA.CMD, the newly updated data base is retained on disk and a backup copy on tape is made with one of the PIP command files SAMTØ.CMD or SAMT1.CMD. And cards are made from the file PUNCH.ZIP by using the utility program CRDOUT to produce a card-image tape. The "control card" file ASATS.BIS is actually a disk file of control card images. The batch run is initiated by the operator with the batch command: BAT ASATS (ALT) (The alternate mode key instead of carriage return allows MESSAGE card information to appear on the operator console).

The complete text of ASATS.BIS is shown in Figure 4-4.

```
$JOB/NAME=ASATS14/LIMIT=999/MCR
$MESSAGE ASATS BATCH STREAM VERSION 14 (14 JUNE 1977).
$MESSAGE FIXED TO SORT PUNCHED CARDS INTO ORDER BY TYPE.
$MESSAGE BE SURE YOU HAVE:
$MESSAGE                    READ IN CARD DECK.
$MESSAGE                 ( WITH:      PIP @UP     )
$MESSAGE IN CASE OF TROUBLE, CALL:
$MESSAGE      JOE EVERETTE 333-6311 (DAYS) OR 554-3660 (NIGHTS)
$MESSAGE OR   DAVE SMITH   333-6311 (DAYS) OR 482-0644 (NIGHTS)
$MESSAGE OR   JOHN COON    483-6427 (DAYS) OR 481-0339 (NIGHTS)
$MESSAGE IF CARDS NOT O.K., ABORT THIS RUN AND RESTART
$MESSAGE WITH ANOTHER BAT ASATS AFTER READING CARDS.
$MCR PIP DUM.TES=ENDFILE
$MCR PIP LP:=*.TES;*/LI
$MESSAGE/WAIT NOW, TYPE IN CON(CR) TO CONTINUE, OR ABO(CR) TO ABORT.
$MCR PIP LP:=*.TES;*/LI
$MESSAGE START PREPROCESSOR
$ ! CLEAN UP FILES
$MCR PIP
DUMY.TES=ENDFILE
*.TES;*/DE
$ ! STEP 1 READS PPF11.CRE AND WRITES OUT THE FOLLOWING:
$ !   PPF142 AND PPF143 (PHASE 2 AND 3 FOR LATER PROCESSING)
$ !   OTHER FILES TO LINE PRINTER
$ !  SORT OPERATION WILL ALWAYS HAVE INPUT AND ALWAYS PUT OUT
$ !  SOMETHING IF IT RUNS SUCCESSFULLY.
$RUN STEP1.TSK
$MESSAGE START SORT
$MCR SRT PPF242.TES=PPF142.TES/SIZE:80,DLSPEC.SOR
$ !  CAN NOW SAVE SPACE BY REMOVING INPUT TO THAT SORT.
$MCR PIP PPF142.TES;1/DE
$MCR SRT PPF243.TES=PPF143.TES/SIZE:80,DLSPEC.SOR
$MCR PIP PPF143.TES;1/DE
$ ! SET UP NOW FOR STEP3.
$ ! STEP 3 WRITES TWO FILES FOR EACH LACIE PHASE:
$ !  PPF33(2 AND 3) OF NON-*, 2, 3 CARDS
$ !  PPF33(2 AND 3) OF *, 2, 3 CARDS
$ ! ALL FOUR MUST EXIST IF NO MESSAGES ARE TO BE GIVEN.
$MCR PIP
PPF332.TES=ENDFILE
PPF333.TES=ENDFILE
PPF352.TES=ENDFILE
PPF353.TES=ENDFILE
$RUN STEP3.TSK
$MCR PIP PPF242.TES;*/DE
$MCR PIP PPF243.TES;*/DE
```

Figure 4-4.—The Batch Run Control Card File, ASATS.BIS.

```
$MESSAGE STEP 3 OF PREPROCESSOR FINISHED
$ ! BEFORE DOING THE SORT, PUT UP A DUMMY OUTPUT FILE FOR IT.
$MCR PIP PPF42P.TES=ENDFILE
$MCR SRT PPF42P.TES=PPF333.TES/SIZE:80/KEYS:CN4,4:CN1,80
$ ! NOW CLEAN UP INPUT FILES TO THAT SORT, AND
$ !  UP DUMMY OUTPUTS FOR STEP 5.
$MCR PIP
PPF333.TES;*/DE
PPF53P.TES=ENDFILE
PPF55P.TES=ENDFILE
PPF57P.TES=ENDFILE
$RUN STEP5.TSK
$MCR PIP PPF42P.TES;*/DE
$MESSAGE STEP 5 FINISHED FOR PHASE 3
$ ! NOW SORT THE *,2,3 NON-SETS BACK INTO CARD-TYPE ORDER.
$ ! FIRST, PUT UP A DUMMY SORTED OUTPUT FILE.
$MCR PIP PPF653.TES=ENDFILE
$MCR SRT PPF653.TES=PPF55P.TES/SIZE:80/KEYS:CN1,80
$ ! NOW DELETE INPUT TO THAT SORT AND CONCATENATE ALL THE UPDATE FILES.
$MCR PIP
LP:=PPF57P.TES
PPF57P.TES;*/DE
PPF55P.TES;*/DE
PHASE3.TES=PPF53P.TES/RE
PHASE3.TES=PPF653.TES/AP
PHASE3.TES=PPF353.TES/AP
PPF53P.TES=ENDFILE
PPF53P.TES;*/DE
PPF653.TES;*/DE
PPF353.TES;*/DE
PPF42P.TES=ENDFILE
$MCR SRT PPF42P.TES=PPF332.TES/SIZE:80/KEYS:CN4,4:CN1,80
$ ! NOW CLEAN UP INPUT TO THAT SORT AND GIVE DUMMY OUTPUT FOR STEP 5.
$MCR PIP
PPF332.TES;*/DE
PPF53P.TES=ENDFILE
PPF55P.TES=ENDFILE
PPF57P.TES=ENDFILE
$RUN STEP5.TSK
$MCR PIP PPF42P.TES;*/DE
$MESSAGE STEP 5 FINISHED FOR PHASE 2.
$ ! SORT *,2,3 NON-SETS BACK INTO CARD-TYPE ORDER.
$ ! FIRST, CREATE A DUMMY SORTED FILE.
$MCR PIP PPF652.TES=ENDFILE
$MCR SRT PPF652.TES=PPF55P.TES/SIZE:80/KEYS:CN1,80
$MCR SRT PPF652.TES=PPF55P.TES/SIZE:80/KEYS:CN1,80
```

Figure 4-4.—The Batch Run Control Card File, ASATS.BIS. (cont)

```
$ ! DELETE INPUT TO THAT SORT,  CONCATENATE UPDATE FILES,
SMCR PIP
PPF55P.TES;*/DE
LP:=PPF57P.TES
PPF57P.TES;*/DE
PHASE2.TES=PPF53P.TES/RE
PHASE2.TES=PPF652.TES/AP
PHASE2.TES=PPF352.TES/AP
PPF53P.TES=ENDFILE
PPF53P.TES;*/DE
PPF652.TES;*/DE
PPF352.TES;*/DE
$ ! ALL FILES ARE NOW CLEANED UP AND UPDATES ARE IN 2 FILES,
$ !  SEPARATED BY PHASE; PHASE3.TES AND PHASE2.TES.
SMESSAGE PHASE 3 UPDATES WILL NOW BEGIN.
SMCR PIP
PPF353.TES=PHASE3.TES/RE
FOR012.DAT=ENDFILE
FOR012.DAT;*/DE
FOR012.DAT=ENDFILE
FOR007.DAT=ENDFILE
FOR007.DAT;*/DE
SMESSAGE THIS IS YOUR LAST CHANCE TO STOP UPDATES (CON OR ABO).
SMCR PIP LP:=*.TES;*/LI
SMESSAGE/WAIT (IF YOU GO PAST THIS POINT, YOU CANNOT RESTART).
SMCR PIP LP:=*.TES;*/LI
SRUN ASATS.TSK
SMCR PIP LQ.TES=*.*;*/LI
SMESSAGE PHASE 3 UPDATES COMPLETED.
SMCR PIP
RM2.POS=ENDFILE
RM2.POS;*/DE
RM2.POS=FOR012.DAT
DUM.ZIP=ENDFILE
*.ZIP;*/DE
SMESSAGE PREPARE OUTPUT REPORTS.
SRUN POSTP.TSK
SMCR PIP CARDS.TES=PUNCH.ZIP/RE
SMCR SRT PUNCH.ZIP=CARDS.TES/SIZE:80/KEYS:CN1,80
SMCR PIP LQ.TES=*.*;*/LI
SMCR PIP PUNCH.ZIP=DATE.COM/AP
SMCR PIP LP:=*.ZIP;*
$ ! START THE OTHER DAILY REPORTS.
SMCR PIP UNITS.SAT;20=BATCH.SAT
SMCR PIP BAT.COM=ENDFILE
SMCR PIP BAT.COM;*/DE
SMCR PIP BAT.COM=REP.COM;1/RE
SRUN RIMS1.TSK
```

Figure 4-4.—The Batch Run Control Card File, ASATS.BIS. (cont)

```
$MCR PIP LQ.TES=*.*;*/LI
$MCR PIP DUM.SHI=EN$FILE
$MCR PIP *.SHI;*/DE
$MCR PIP BAT.COM;*/$F
$MCR PIP BAT.COM=REP.COM;2/RF
$RUN RIMS2.TSK
$MCR PIP LQ.TES=*.*;*/LI
$MCR PIP DUM.SHI=EN$FILE
$MCR PIP *.SHI;*/DE
$MCR PIP BAT.COM;*/$F
$MCR PIP BAT.COM=REP.COM;3/RF
$RUN RIMS3.TSK
$MCR PIP LQ.TES=*.*;*/LI
$MCR PIP DUM.SHI=EN$FILE
$MCR PIP *.SHI;*/DE
$MCR PIP BAT.COM;*/$F
$MCR PIP BAT.COM=REP.COM;4/RF
$RUN RIMS4.TSK
$MCR PIP LQ.TES=*.*;*/LI
$MCR PIP DUM.SHI=EN$FILE
$MCR PIP *.SHI;*/DE
$MCR PIP BAT.COM;*/$F
$MCR PIP BAT.COM=REP.COM;5/RF
$RUN RIMS5.TSK
$MCR PIP LQ.TES=*.*;*/LI
$MCR PIP DUM.SHI=EN$FILE
$MCR PIP *.SHI;*/DE
$MCR PIP BAT.COM;*/$F
$MCR PIP UNITS.SAT;20/DE
$MCR PIP LP:=PHASE2'*';*/LI
$MESSAGE ***********************************************
$MESSAGE * END OF ASATS PHASE 3 BATCH UPDATES AND REPORTS  *
$MESSAGE * REMEMBER TO:                                    *
$MESSAGE *              MAKE CARDS (USE CRDOUT ON THE      *
$MESSAGE *                          FILE PUNCH.ZIP)        *
$MESSAGE *       AN$    MAKE LABELS (LOAD LABELS INTO      *
$MESSAGE *                          PRINTER AND DO PIP @LA  ) *
$MESSAGE *       AN$    SAVE [210,004] ONTO TAPE           *
$MESSAGE *              ( HEL [5,5]                        *
$MESSAGE *                INI MT0:P3 DATE/UIC=[210,004]    *
$MESSAGE *                MOU MT0:/OVR                     *
$MESSAGE *                HEL [210,004]                    *
$MESSAGE *                PIP @SAMT0:                      *
$MESSAGE *          $ASTEN THE DIRECTORY TO THE TAPE )     *
$MESSAGE * THIS IS THE END OF THE ASATS PHASE 3 BATCH RUN. *
$MESSAGE ***********************************************
$MCR PIP LP:=*.TES;*/LI
$MESSAGE NOW LOAD 5-PART PAPER INTO THE PRINTER AND
$MESSAGE/WAIT TYPE $N CON (CR) TO PRINT REPORTS.
$MCR PIP LP:=*.TFS;*/LI
$EOJ
```

Figure 4-4.— The Batch Run Control Card File, ASATS.BIS. (Concluded)

4.3 SORT UTILITY SPECIFICATION FILE DLSPEC. SOR

This file controls the sort utility (SRT) for the two sorts of update card files which occur between step 1 and step 3 of the preprocessor. The purpose of the sort is to put the cards into order by card type (column 2) for proper order of processing of two or more updates on a single segment or acquisition, and to be sure that duplicate cards are detected (because after the sort, duplicates will be together in the file). The ordering of card tapes is effected by a "force" (F in column 7 of the description of the type field) which, in effect, changes the collating sequence only in that field. Other fields are sorted on the standard ASCII collating sequence.

The text of DLSPEC.SOR is shown in Figure 4-5.

```
00 HSORTR     90A          X  90  HEADER CARD
01 FFC        2D*
02 FFC        2*2=
03 FFC        223=
04 FFC        234=
05 FFC        245=
06 FFC        256=
07 FFC        267=
08 FFC        2AB=
09 FFC        279=
10 FFC        28B=
11 FFC        29G=
12 FFC        2GH=
13 FFC        2HI=
14 FFC        2TJ=
15 FFC        2NK=
16 FFC        2JM=
17 FFC        2KN=
18 FFC        2MQ=
19 FFC        2TT=
20 FFC        2XU=
21 FFC        2UX=
22 FNC     3  80
23 FDC     1  80
```

Figure 4-5.— The Sort (SRT) Specification File, DLSPEC.SOR.

4.4 ASATS/RIMS COMMAND FILES

An ASATS/RIMS command file is a series of RIMS commands. Report files tend to be repetitive. The most frequently used command files represent a query report situation where a lengthy series of commands would otherwise have to be manually input interactively each time the specific report is desired. The command files, therefore, provide for consistency of reports and, perhaps more importantly, for convenience to request an already debugged stream of commands to the ASATS data base. The command file approach also offers convenience in that the files may be used as often as required in either batch or interactive mode. Other command files may be used for controlling processing sequences such as the Update Control File (RM4.COM).

4.4.1 RM4.COM – AN UPDATE CONTROL FILE

The following is a listing of the standard card file update control commands:

```
BEDB01XX                        UP
RF0,12,RED.DAT                  UP
RF12,12                         UP
RE                              UP
RF0,12,FOR012.DAT               UP
RF0,11,DB01PPF353.TRS           UP
RF11,11                         UP
UP
UP
UP
UP
UP
UP
UP
UP
UP
UP
UP
UP
UP
UP
UP
UP
```

NOTE: The RE and UP commands as used herein are special ASATS commands to read and interpret the Procedure 1/normal ASATS processing type code and to update the ASATS data base. Thus, the RE and UP commands as used here should not be confused with the RIMS commands of the same mnemonic; i.e., ... RE for "Restructure" and UP for "Unpost".

4.4.2 OP13.COM – OPERATIONS STATUS SUMMARY OF SEGMENTS IN THE DAPTS DATA BASE REPORT COMMAND FILE

The following list represents the standard report generating commands to produce the frequently requested OPS Status Summary of Segments in the DAPTS DB:

```
BE
SKCOUNTR@COUNTRZ
RF12,10
HD1,1
HD1,                                                    LACIE PHASE III
HD2,                                          OPS STATUS SUMMARY OF SEGS IN DAP
TS DB
RF0,12,DB0:DATE.COM
RF13,12
HD1,0
HD2, SEG                          G W        BW 1  BW 1 BW 2  BW 2 BW 3  BW
 3 BW 4  BW 4                       LAST
HD2, NO.  CNTRY  RG ZONE STR  D V PC TY OPEN CLOSE OPEN CLOSE OPEN CLO
SE OPEN CLOSE TOPO CROP ANCL  CHANGE
HD1,
PF1,69
EN
```

This report command file additionally illustrates use of the report header commands available to the user plus demonstrating how another command file may be called to become an integral part of this command stream (see line RFØ,12,DBØ:DATE.COM).

4.4.3 OP23.COM – OPERATIONS STATUS SUMMARY OF ACQUISITIONS COMMAND FILE

The following list represents the standard report generating commands to produce the frequently requested OPS Status Summary of Acquisition report (similar to the report commands of par. 4.4.2).

```
BE
SKBW      @BW     @
RF12,10
HD1,1                                                   LACIE PHASE II
HD2,                                          OPS STATUS SUMMARY OF AC
QUISITIONS
RF0,12,DB0:DATE.COM
RF13,12
HD1,0
HD2, SEG  ACQ   R W   TAPE  F
                R           W      CAM CA
HD2, NO   DATE W V   NO    F RG PC GSFC        COMMENT          R    G
 H    I    J   C  K    M   C  X   BP  TG LSD
JF1,70
EN
```

4.4.4 POLIST.COM – PACKET ORDER LIST COMMAND FILE

The following partial list illustrates a quite lengthy, but mostly repetitive type of report generating commands to produce the daily requested packet order list. This redundancy is fairly obvious in the report header commands; however, note that the data set isolation command (SKPC 2, SKPC 3, etc.) changes with each iteration. The purpose here is to provide standardized format for a group of reports which operate sequentially upon different data sets.

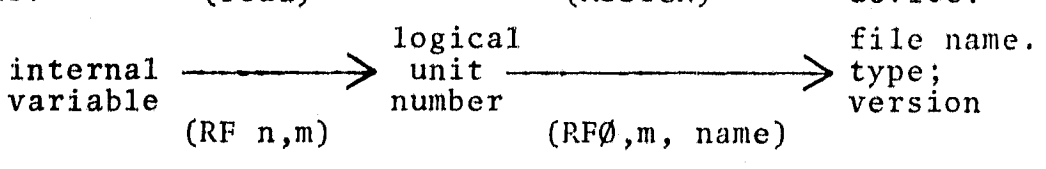
```
BE
RF12,10
HD2,1                                                    DAILY PACKET ORDER L
IST
RF0,12,DATE.COM
RF13,12
HD1,
HD1, PHASE III        US       PC2
HD1,
HD1,
HD2, ORD SEG    LPI   ACQ    REC   ZONE   STR   B   W   CNT     LAST    DEL
TX    REC   CAMS/LPDL   COMMENT
HD2,         NO          DATE                        W   V           CHNG          D
ATE
RF12,7
SKPC      2
GC1
SN2,AICOMP,NE,     ,PACKRF,EQ,     ,
RF12,10
JF3,90
SC3,46,    TOTAL ACQUISITIONS
BE
RF12,10
HD2,1                                                    DAILY PACKET ORDER L
IST
RF0,12,DATE.COM
RF13,12
HD1,
HD1, PHASE III        US       PC3
HD1,
HD1,
HD2, ORD SEG    LPI   ACQ    REC   ZONE   STR   B   W   CNT     LAST    DEL
TX    REC   CAMS/LPDL   COMMENT
HD2,         NO          DATE                        W   V           CHNG          D
ATE
RF12,7
SKPC      3
GC1
SN2,AICOMP,NE,     ,PACKRF,EQ,     ,
RF12,10
JF3,90
SC3,46,    TOTAL ACQUISITIONS
```

5. ASATS/RIMS FILE USAGE

This chapter describes the data files used in the daily operations of the ASATS interactive and batch mode systems. The control files which control sequences of utility or batch operations have already been described in chapter 4. Command files for building the system (indirect task builder files, etc.) will be described in chapter 6.

5.1 THE RELATION OF INTERNAL FILE DESIGNATIONS TO EXTERNAL UNITS AND FILE NAMES

Most of the tasks in the ASATS/RIMS system have a flexible relation between internal unit designations and external file names that can be varied at load time or during a run, rather than being fixed by default FORTRAN complier and/or task builder assignments. This generalized relation is built from two independent relations on three independent sets of designators: a relation of fixed internal designations to external logical unit numbers, and another relation established between those logical unit numbers and system file names (device: name. type; version). The first relation is established by an array of logical unit numbers read by the program at the beginning of a run and possibly changed again during the run. The second relation is established by a call to the system subroutine ASSIGN, and can also occur either at the beginning of a run or during the run. The complete relation is the composition of the two relations:



The RF command to RIMS can be used in its two different forms to alter either relation, as shown above. The relations are both established at the beginning of a RIMS run by the subroutine UNITS, which always reads a file UNITS.SAT. That file

contains one record of logical unit numbers to establish the first relation, and five more records that give a unit number and a file name to establish the second relation. One version of the file UNITS.SAT is permanently on the disk and will be picked up for interactive runs to establish command file input from the user terminal TI:. A batch run will precede the RUN of RIMS by a copy operation from the file BATCH.SAT to a new version of UNITS.SAT. The batch run of RIMS will therefore supersede the usual interactive command file from the TI: with an assignment to the disk command file. That file can contain RF commands which will further alter either relation. It should be noted that the form "RFØ,m,name" will close any file already associated with logical unit m and re-open m. This means that file m will be positioned at the beginning of the new file. This provides a means for using one command file several times. It could also cause an infinite loop if the command file re-opens itself. It also allows run-time manipulation of command or data files. A file of commands could conceivably be written as a report file, closed and re-opened as a new version of a command file. Or, a report file could be created, closed, re-opened as a data input file, and used to update the data base.

The tasks of the preprocessor and the postprocessor and the ASATS utility tasks CONTAP and JJ were all built with similar file-assignment capabilities. Each reads its own file, with a file type "IUN", at the beginning of a run to establish both logical unit numbers and file names. These tasks do not change file assignments later in execution.

5.2 FILE TYPES USED IN THE ASATS SYSTEM

Table 5-1 lists the file type mnemonics used in the ASATS system. Some of the file types are standard RSX-11-D type conventions.

5.3 BATCH RUN DATA FILES

The data files used in the preprocessor are named with 5- or 6-character names. The first 3 characters are always "PPF" and the next two digits indicate, respectively, the step of the preprocessor which uses the file and the logical unit number in the program. A final digit "2" or "3" is used if there are two similar files for the two LACIE phases. Sometimes the final digit is replaced by the letter "P" to indicate that one file name is used in two runs of a program for the two LACIE phases.

The contents of these files are described in table 5-2.

TABLE 5-1.– MEANINGS OF FILE TYPES IN ASATS SYSTEM

File type	Usage
.(blank);	1. Report "starter" files, usually containing RIMS commands: BE, password, and an RF to the command file (same name, type COM) that produces the report. 2. Also used as the file type for ENDFILE and BLANK (two empty files used to create dummy files).
.BIS;	Batch input stream
.CMD;	System utility indirect command files
.COM;	RIMS "canned" command files
.CRE;	A card-image file from the card reader
.DAT;	Data files (FORTRAN default type)
.IUN;	Unit-name assignment files (as described in section 5.1)
.POS;	The input to the postprocessor
.Rn; (.R1;, .R2;, •••)	Data base files
.SAT;	Unit assignment files for RIMS (see section 5.1)
.SOR;	A file for the system sort utility (SRT)
.TES;	Update card image and listing files in the preprocessor
.TSK;	Task-built executable modules
.ZIP;	Output print label, or punch files from the postprocessor

TABLE 5-2.– CONTENTS OF BATCH RUN FILES

File Name	Content
PPF11.CRE	The input to step 1 of the Preprocessor.
PPF13.TES	The unsorted listing of input cards with record count.
PPF142.TES	The LACIE Phase II cards, with a Q card.
PPF143.TES	The LACIE Phase III cards, with a Q card.
PPF16.TES	Input cards with invalid type found by step 1 of the Preprocessor.
PPF17.TES	Cards found by step 1 of the Preprocessor to have invalid LACIE Phase.
PPF18.TES	Cards found by step 1 of the Preprocessor to have some non-blank character in columns which step 1 checks for blanks. The images fed to RIMS will be forced to blank in those columns.
PPF19.TES	A file produced by step 1 of the Preprocessor, containing counts of cards and other statistics from the first reading of the update cards.
PPF242.TES	Input for step 3 of the Preprocessor (Phase 2)
PPF243.TES	Input for step 3 of the Preprocessor (Phase 3)
PPF332.TES	The output from step 3. It contains all the *, 2, and 3 cards with LACIE Phase = 2.
PPF333.TES	The output from step 3. It contains all *, 2, and 3 cards with LACIE Phase = 3.
PPF352.TES	All Non-*, 2, or 3 cards put out by step 3 of the Preprocessor for LPI = 2.
PPF353.TES	All Non-*, 2, nor 3 cards put out by step 3 of the Preprocessor for LPI = 3.

TABLE 5-2.– Concluded.

File name	Content
PPF34.TES	A listing of all input cards sorted by type (or as received by step 3 of the Preprocessor.)
PPF38.TES	A listing of invalid duplicates found by step 3 of the ASATS Preprocessor.
PPF42P.TES	The input file for step 5.
PPF53P.TES	Output from step 5. It is the file of complete sets of *, 2, 3 for a single segment.
PPF55P.TES	Output from step 5 of the Preprocessor. It is the file of all *, 2, and 3 cards which do not fall into complete sets for a segment.
PPF57P.TES	The listing file of step 5 of the Preprocessor.
PPF653.TES	Non-set *, 2, 3 found by steps 5 and sorted back into card type order: * cards, then 2 cards, then 3 cards.

6. ASATS EXECUTABLE TASK DESCRIPTIONS

This chapter describes ASATS program sets at the task level. There are three types of tasks in the ASATS system: data base manipulation tasks, batch run edit, audit and report formatting tasks (pre- and post-processor), and utility tasks used to build or pack the data base. The tasks are listed by type in table 6-1.

6.1 TASK BUILDER COMMAND AND OVERLAY DESCRIPTION FILES

The ASATS tasks are each built with an indirect command file for the RSX-11D Task Builder. Each command file has the same name as the task, and file type ".CMD.". Two of the ASATS tasks, ASATS.TSK and RIMS.TSK are built in overlay form, and the respective command files refer to Overlay Description files ASATS.ODL and RIMS.ODL. Listings of .CMD and .ODL files are included in Attachment A to this document.

6.2 ASATS TASK EXECUTION INSTRUCTIONS

Most of the ASATS tasks are well documented in terms of user application in the ASATS and RIMS user guides. Two additional tasks were created expressly for ASATS and have no direct relation to RIMS. These tasks were used to manipulate files prior to the initial load of the data base transferred to us on tape from COMSHARE. The tasks have been retained because there is a continuing need for utilities to read foreign tapes and break down large sequential files into smaller pieces. These two tasks are described on the following pages.

TABLE 6-1.— ASATS TASK FUNCTIONS

Task type	Task name	Function
Data Base Management	ASATS.TSK	Updates the data base in the batch run. (a subset of RIMS)
	RIMS.TSK	Interactive data base update, and report generation in both interactive and batch modes.
Edit and separate updates; produce report listings	STEP1.TSK	Makes first pass over the update cards and separates them by LACIE Phase
	STEP3.TSK	Separates *, 2, and 3 card types from all other types; inserts dates from the Q card.
	STEP5.TSK	Finds complete sets of *, 2, 3 cards for a segment and separates the sets from individual *, 2, 3 update cards.
	POSTP.TSK	Separates output from the update task into separate listings and punch files.
Utilities	CONTAP.TSK	Reformat a sequential data base file in preparation for loading by RIMS.
	CR. TSK	Create direct-access files for use as a data base.
	CREATE.TSK	Initial creation of a data base.
	JJ.TSK	Copies selected records from one sequential file to another.
	NEWFILE.TSK	Builds new files for an empty data base.
	PF1.TSK	Pack data base files .R1 and .R2.
	PF2.TSK	Pack data base files .R3 and .R4.

TASK DESCRIPTION

Task Name: CONTAP

Purpose: To read and reformat a sequential file to prepare it for loading into a data base. The reformatting is generalized, table driven, and programmable.

Setup Required Before Run:

- Files Required: CONTAP.IUN (which specifies other file names)
  Control table file (as unit 3 in CONTAP.IUN)
  Input and output files written by the control file.
- Other: Can mount a tape as an input file, if desired.

Run Instructions:

- Interactive Mode: The user must make available a reformat control table in one file to be read by CONTAP and put the name of that file into CONTAP.IUN along with the names of any input or output files to be processed by the reformat table. The reformat table is a sequence built from the following operations:

  a. Read a file to the input buffer and skip next table row unless end-of-file.

  b. Move field from input buffer to output buffer.

  c. Convert characters from EBCDIC to ASCII.

  d. Check field for non-numeric and skip next row of table if numeric.

e. Write out the output buffer.

f. Jump to row N of the table.

g. Stop.

There are several other possible operations that were specifically built for conversion of data from COMSHARE formats to the new PDP-11 ASATS formats. A detailed description of the parameters required in each table row is found in the program listing.

TASK DESCRIPTION

Task Name: JJ

Purpose: To break a sequential file into smaller files.

Setup Required Before Run:

- Files Required: JJ.IUN, which must specify an input file name for unit 2 and an output file name for unit 1.

Run Instructions:

- Interactive Mode:

```
MCR>RUN JJ    ALT
ENTER START AND END RECORD NUMBERS
AND RECORD LENGTH IN 315 FORMAT
____1___12___80
(CZ)
JJ--STOP
```

(The above run would copy the first 12 records from the input file to the output file with a record length of 80 characters).

Output Files: The output file is selected by the JJ.IUN unit assignment file.

7. NEW AND MODIFIED PROGRAMS

This chapter contains the as-built design details of individual programs (main programs, functions, and subroutines). Included are all new programs built especially for ASATS, plus programs from the RIMS system which had to be modified for the ASATS application.

For documentation of those RIMS programs which have remained unchanged, refer to the RIMS Maintenance Document (LEC-9566, October 1976).

Complete listings of the ASATS program source files are given in Attachment B to this document.

Name: ABORT (preprocessor step 1)

Purpose: If two different Q cards exist, then this routine will abort the job.

Linkage:

- Calling sequence: CALL ABORT
- Common blocks used: UNITS
- Subroutines or functions used: EXIT
- Files used: Logical Units 4, 5, 10

Input Description: Unit numbers from array IUN - logical unit numbers for the listing file and output data files.

Output Description: Update card files 4 and 5 are erased.

Process Description: A check of the count of Q card images is made. If Q-type count is not exactly one, the job will abort sending ABORT messages to the operator's console and the line printer file. Units 4 and 5 get rewound and have a file mark written into them. This prevents making any updates to the data base.

Name:	ABORT (preprocessor step 3)
Purpose:	To prevent making a data base update when the update card file is pathological.
Linkage:	• Calling sequence: CALL ABORT • Common blocks used: UNITS • Subroutines or functions used: EXIT • Files used: Unit 7
Input Description:	Listing file unit number in IUN(7)
Output Description:	A banner message on the print file and a message to the operator that the update run is to be aborted.
Process Description:	A FORTRAN write statement and a PAUSE message.

Name: ACCNO

Purpose: To produce a binary accession number (record ID) corresponding to a character string.

Linkage:

- Calling sequence:

  number = ACCNO (string, start, length)

  where

  number is the output value,

  string is a character string array,

  start is the starting character position, and

  length is the number of characters to be converted

- Common blocks used: None
- Subroutines or functions used: INDEX, INPARM
- Files used: None

Input Description: Start and length should be typed INTEGER*4. String should contain a string of characters that is either: (a) two strings of digits separated by an "@", or (b) a single string or digits terminated by either the end of string or by a non-numeric character other than "@".

Output Description: The output is in double-word (INTEGER*4) form and is, for a single number, the binary value of the number. For the double-number form of input, int 1 @ int 2, the value of int 1 is put in the second output word, with value of int 2 in the first word.

format	output word 2	output word 1
integer	zero (or overflow)	value of integer
int 1 @ int 2	value of int 1	value of int 2

Process Description: Look for the "@", and then convert one or two integer parts.

Name: APSEL

Purpose: Same as SELECT, except that an empty set will be selected independent of the ZZ (retain empty set flag) status.

Linkage:

- Calling sequence: CALL APSEL
- Common blocks used: SYSCOM
- Subroutines or functions used: LOCATE
- Files used: U(3), U(4), U(7)

Input Description: See SELECT in the RIMS Maintenance Document

Output Description: See SELECT

Process Description: See SELECT

Name: ASN

Purpose: Assigns file RM4.COM to a specified unit. Note: Subroutine exists in order to provide better control of position of system subroutines in overlays.

Linkage:

- Calling sequence: CALL ASN (FN)
- Common blocks used: None
- Subroutines or functions used: None
- Files used: Unit specified on input

Input Description: FN contains the unit assignment on file to be assigned (RM4.COM).

Output Description: None

Process Description: File RM4.COM is assigned to specified unit.

Name: AUDATE

Purpose: Updates data base from set of input cards. Specific update operations are a function of card type (specified in second character of each card), data base format, and the input format. The input format and data base to be used are also a function of the card type.

Linkage:

- Calling sequence: CALL AUDATE
- Common blocks used: SYSCOM, SY2COM, PDT
- Subroutines or functions used: GETREC, LODFMT, AUREC, CHFLD, APSINT, AUPOST, APSTUP, SUBSTR, KOMSTR, APSCNT, SETOUT, XXOUT, ENDAT, GETCLD
- Tables used: Processing Description Table (PDT) which contains card type versus type of record ID generation and the category type table. The bio-window table from segment record.
- Files used: Units 1, 2, 3, and 4 are updated. Units 5, 6, 9, and 10 are scratch files. ASATS cards are on the RIMS data file. The RIMS message file is used for ASATS postprocessor data.

Input Description:

- Command: Processing is begun by a UP command
- Status and tracking input cards: Any of the 21 types of ASATS update cards (except Q cards) are processed sequentially until an EOF.

Input Description: (continued)

- EOF: Processing of an input file is ended by a blank card (inserted by the preprocessor) or by actual end-of-file.

Output Description:

Besides updating the ASATS data base the following information is recorded sequentially on a file.

- Rejected input cards
  - Required DAPTS record does not exist (for *, 2, 3, 4, 5, and 6 cards)
  - Required FLOCON record does not exist
  - FLOCON record has not reached required state for particular type of card.
- Accepted input cards which create new DAPTS records.
- Punch cards

Process Description:

The required processing is a function of card type. Card types are categorized as follows:

- Category 1 – card type 2
- Category 2 – card types *, 2, 4, 5, 6 and T
- Category 3 – card type 3
- Category 4 – card type B
- Category 5 – other card types
- Category 6 - N card

A generalized function for adding new records and updating existing records will exist. This function, which is driven by

input formats, data base formats, and card type, will add or modify the specified record. The general steps of processing input cards are as follows.

- Read input card
- Generate record ID (See table 7-2)
- Generate external (input) format ID from table
- Retrieve record
- Retrieve formats
- Either add or update record (or both in case of "N" card)
- Output card image reflecting success or error

Figure 3-7 depicts the flow of this process and variations dependent upon category of card type.

Table 7-1 indicates the data used for generating a record depending on input category. The input format is a function of the card type.

The type of operation, an add or modify, to be performed is a function of the category for the record type and whether or not a record already exists. The input format for the card type identifies the fields it updates and the field's data types. Table 3-5 describes the processing for field types on input. The processing of individual fields is transparent to this routine (it

Process Description: (continued)

is performed by AUREC).

If an error condition occurs when processing an input card, an error type is put in column 2 when the image is written to the message file. The input card images for new DAPTS records are also written to the message file; column 2 for them is the logical unit number for the DAPTS record file.

Additional processing required by category 3 is the selection of FLOCON records of the same segment and the updating of their biowindow fields.

TABLE 7-1.– PROCESSING DESCRIPTION AND FORMATS FOR ASATS UPDATES

Card type	Category	Method for Gen. RID	Input format number
Z (*,2,3)	1	1	21 (52 for ADD)
*	2	1	21
2	2	1	22
3	3	1	23
4	2	1	24
5	2	1	25
6	2	1	26
B	4	2	27
G	5	2	28
H	5	2	29
I	5	2	30
J	5	2	31
K	5	2	32
M	5	2	33
X	5	2	34
U	5	2	36
7	5	2	37
8	5	2	36
9		2	38
N	6	2	40
T	6	2	41

TABLE 7-2.-- CARD TYPE VERSUS RECORD ID GENERATION TABLE.

Card Type	Method of Generating Record ID
Z, 2, 3	Segment number
*	Segment number
2	Segment number
3	Segment number
4	Segment number
5	Segment number
6	Segment number
T	Segment number
B	Segment number and Acq. date
Others	Segment number and Acq. date

Name: AUREC

Purpose: To either add or update a record to the data base.

Linkage:

- Calling sequence: CALL AUREC (J, IFUN, STAT)
- Common blocks used: SYSCOM, SY2COM, UPDCOM
- Subroutines or functions used: LMVTAB, APSTUP, TFORM, ADDR, REPR
- Files used: Units 9 and 10 for auto posting.

Input Description: IFUN = 1 indicates ADD, IFUN = 2, indicates update

SY2COM contains the record to be added or modified

J = record ID

STAT = status return

Output Description: Data base is updated or modified
Key changes are on Unit 9 and 10
Status reflects results of operation
- status = 0
- status = 1

Process Description: The system move table is updated

Fields are transformed from input array to output array according to move table

Fields are posted if required

Fields are unposted if required

Process Description: (Continued)

Records are then added or modified using ADDR or REPR

Name: BLKLN (preprocessor step 3)

Purpose: Prints a blank line

Linkage:

- Calling sequence: CALL BLKLN
- Common blocks used: UNITS, LINES
- Subroutines or functions used: None
- Files used: UNIT = 7

Input Description: Unit number IU37
Line counter LINE

Output Description: Writes one blank character to unit 7, increments the line counter LINE

Process Description: Write unit 7 and increment LINE.

Name:	BLNKCK (preprocessor, step 1)
Purpose:	To check update cards for nonblank characters and force blanks in columns that will be used for flags.
Linkage:	• Calling sequence: CALL BLNKCK • Common blocks used: UNITS, IMAGES, BCHK • Subroutines or functions used: None • Files used: Logical unit 9
Input Description:	Update card image in IMG array; a list of card columns in array ICOL
Output Description:	The same card image, with listed columns forced to blanks. A list of cards in which forcing was required on unit 9
Process Description:	Each column in the list ICOL is checked for a blank. If any column is nonblank, the unchanged image is written to a list file, and all columns are then forced blank. A non-blank is not a fatal error of the update.

7-17
67

Name: CARTP (preprocessor step 3)

Purpose: To separate cards with types *, 2, or 3 from all other update cards.

Linkage:

- Calling sequence: CALL CARTP
- Common blocks used: FLAGS, UNITS, LINES, IMAGES
- Subroutines or functions used: PAGE, HDR, BLKLN, COLABL
- Files used: Units 1, 3, 5, 7

Input Description: Card images in array IM1

Current phase's unit number for *, 2, 3 cards in IU33P

Unit number for other card types in IU35P

Output Description: A count of cards of one type in ICTCT. Card images to the listing file (unit 7) and to other units as follows:

Unit	Type	Phase
3	*,2,3	2
4	*,2,3	3
5	Other	2
6	Other	3

Process Description: Appropriate unit numbers for the phase now being processed are selected by card type of the image in the IM1 array. Calls to subroutines provide page and column headers and page number control.

Name: CHAR

Purpose: To convert an integer to a string of ASCII digits.

Linkage:

- Calling sequence: CALL CHAR (string, start, length, value)

  where

  value is the integer to be converted,

  length is the number of digits to be produced,

  start is the leftmost character position in the output string, and

  string is the output string.
- Common blocks used: None
- Subroutines or functions used: None
- Files used: None

Input Description: Value, length, and start should be typed INTEGER*4

Value should contain a non-negative integer in the range $0 \leq \text{value} \leq 2^{31} - 1$

The array string should be long enough to hold start + length - 1 characters

Output Description: A field of length characters is filled with the ASCII codes for the decimal representation of value. Leading zeros are not suppressed. Overflow of the field is not detected: the low-order digits will be given with no error indication.

Processing Description: Standard divide-by-modulus-and-use-remainder conversion algorithm.

Name:	CKTYLP (preprocessor step 1)
Purpose:	To check update cards for type and phase
Linkage:	• Calling sequence: CALL CKTYLP • Common blocks used: IMAGES, VALID, FLAGS, UNITS • Subroutines or functions used: QCHCK • Files used: Logical units 4, 5, 7, 8
Input Description:	Update card image in array IMG
Output Description:	The updated card image list of invalid card type on unit 7 and invalid phase on unit 8. Card images of phase 2 and phase 3 are written on units 4 and 5, respectively.
Process Description:	Each card is initially checked for Q-type in column 2. If Q-type, the card is treated as a special case. If element 2 of the image has an invalid type or phase, the image is written to a list file (units 7 and 8, respectively). All valid phase 2 and phase 3 images are written to output files (units 4 and 5, respectively).

7-21
91

Name: CLOSEL

Purpose: To insert a trailing comma if necessary.

Linkage:

- Calling sequence: CALL CLOSEL (STR, S, L)
- Common blocks used: None
- Subroutines or functions used: LAST
- Files used:

Input Description:

STR - string name
S - start
L - length

Output Description: STR with a trailing comma.

Process Description: Find the last non-blank character and insert a comma as command delimiter.

Name: CMDRI

Purpose: To control the process of the arithmetic command CM.

Linkage:

- Calling sequence: CALL CMDRI
- Common blocks used: SYSCOM, SY2COM
- Subroutines or functions used: INDEX, INPARM, SUBSTR, CMPUTE
- Files used: None

Input Description: User's command string through common block SYSCOM (STR)

Output Description: Mean, standard deviation, count of records used in computation and count of total number of records in a given set. (See output description of subroutine CMPUTE).

Process Description: The following processing is performed.

The command string is parsed.

A format array is set up with the two field names.

Working program (CMPUTE) is called.

Name: CMPUTE

Purpose: To compute mean and standard deviation of the differences of two fields from a set of records.

Linkage:

- Calling sequence: CALL CMPUTE (RID,SET)
- Common blocks used: SYSCOM, SY2COM
- Subroutines or functions used: ADDR, CLOSEP, GETREC, INPARM, LMVTAB, SETIN1, VERIFY, XXIN1, SSORT, LODREC
- Files used: None

Input Description: RID is the record ID for the record where the results of the computation are to be stored.

SET is the set number for which all computations are to be performed.

Output Description: Mean and standard deviation of differences of two fields, count of number of records used in mean computation and the count of the total number of records are stored in the specified record.

Process Description: The following processing is performed.

Get record ID from set
Retrieve record and associated format
Process retrieved record as follows:
Differences of two fields is computed for records in which neither field is blank.

Process Description:
(continued)

Count of number of records used in computation of mean and standard deviation is maintained.

The mean and standard deviation are computed after all records have been processed.

Encode all computed results to alphanumeric.

Call ADDR to store encoded results in the specified record.

Name: COLABL (preprocessor step 3)

Purpose: Provides a printed header giving field identification for the update card listing.

Linkage:

- Calling sequence: CALL COLABL
- Common blocks used: LINES, UNITS
- Subroutines or functions used: BLKLN
- Files used: Logical UNIT = 7

Input Description: LINE (current line number on the page now being printed)

ICOLAB (a flag that indicates whether column labels have already been printed on the current page. The flag is reset when HDR starts a new page)

Output Description: Column headers for the fields of update cards (card type, segment number, etc.)

Process Description: If the flag is reset (ICOLAB = 0) the headers are printed and the flag is set (ICOLAB = 1)

Name: COMSTR

Purpose: To compare two strings (SEE KOMSTR).

Linkage:

- Calling sequence: I = COMSTR (A, S1, L1, B, S2, L2)
- Common blocks used: None
- Subroutine or functions used: None
- Files used: None

Input Description:

A - first string
S1 - start in first
L1 - length in first
A - second string
S2 - start in second
L2 - length in second

Output Description:

-1 - A<B
0 - A=B
+1 - A>B

Process Description: If L1 ≠ L2 the shorter string is considered to be blank filled.

Name:	COUNT (preprocessor step 3)
Purpose:	Puts out a count of the current card type.
Linkage:	• Calling sequence: CALL COUNT (IC) • Common blocks used: LINES, UNITS • Subroutines or functions used: None • Files used: Logical UNIT = 7
Input Description:	The integer card count in parameter IC
Output Description:	On the listing file 7, a 6-digit integer starting in print column 88
Process Description:	A FORTRAN write statement.

Name: CREATE

Purpose: Initial data base load

Linkage:

- Calling sequence: Main program
- Common blocks used: None
- Subroutines or functions used: GETLEN, GETSTR, GETKEY
- Files used: None

Input Description: Depends on application.

Output Description: A RIMS-format data base

Process Description: For a large initial load this is the most efficient method. The three routines used are application dependent, and must be user supplied.

Name: DASHES (preprocessor step 3)

Purpose: Writes out a line of dashes to separate counts from totals.

Linkage:

- Calling sequence: CALL DASHES
- Common blocks used: LINES, UNITS
- Subroutines or functions used: None
- Files used: Unit 7

Input Description: Listing file unit number in IU37

Output Description: 10 hyphens in print columns 84 through 93

Process Description: A FORTRAN formatted write.

Name: DSJFM

Purpose: To control the process of the JF command (display data from two data base levels) and to load the display format.

Linkage:

- Calling sequence: CALL DSJFM
- Common blocks used: SYSCOM, SY2COM
- Subroutines or functions used: INDEX, INPARM, LODFMT, RECTRC
- Files Used: None

Input Description: User's command string through common block SYSCOM (STR)

Output Description: A set of records containing data from two data base levels is displayed.

Process Description: The following processing is performed:

The command string is parsed.

The display format record is loaded by given format ID.

Working program (RESTRC) is called.

Name: ENDSET

Purpose: To enter a set into the status table.

Linkage:

- Calling sequence: CALL ENDSET (HIT, UNIT)
- Common blocks used: SYSCOM
  ALT. ENTRY POINT: MODE
- Subroutines or functions used: NHITS, XXOUT
- Files used:

Input Description: HIT = # items in set
UNIT = logical unit containing set

Output Description: Updated status table.

Process Description: Test for non-empty set and use XXOUT to write set to file 5.

Name: EQUALS (preprocessor step 3)

Purpose: Writes out a line of equal signs to separate count from grand totals.

Linkage:

- Calling sequence: CALL EQUALS
- Common blocks used: LINES, UNITS
- Subroutines or functions used: None
- Files used: Logical UNIT = 7

Input Description: Print file unit number in IU37

Current line number in LINE

Output Description: 10 "=" signs in print columns 84 through 93

Process Description: A FORTRAN formatted write statement.

Name: GETCLD

Purpose: To form a set of children records for a given set.

Linkage:

- Calling sequence: CALL GETCLD (SET)
- Common blocks used: SYSCOM
- Subroutines of functions used: SETIN1, SETOUT, XXIN1, XXOUT, ENDSET, LOCREC, GET
- Files used: Input set on Unit (5) or Unit (3) and output set is on Unit (5).

Input Description: Set contains the parent set number.

Output Description: The children set description is placed in the next available set of the status table. Resulting set is on Unit 5.

Process Description:

1. Initialize input/output files, counters
2. Get next record ID from input, end
3. Error check
4. Position data base unit 2 to parent
5. Follow logical sequential read in unit 2, writing children record ID's in target set until left half of word does not match parent record ID.

7. Close output file, update status table, TABNO, etc.

Name: GETLEN

Purpose: To return, via KEYLEN, the length of the keys in words.

Linkage:

- Calling sequence: CALL GETLEN (KEYLEN)
- Common blocks used: Unknown
- Subroutines or functions used: Unknown
- Files used: Unknown

Input Description: (As desired by the user of CREATE)

Output Description: KEYLEN

Process Description: (To be supplied by the user)

Name: GETPAR

Purpose: To form a set of parent records for a given set.

Linkage:

- Calling sequence: CALL GETPAR(SET)
- Common blocks used: SYSCOM
- Subroutines or functions used: SETIN1, SETOUT, XXIN1, XXOUT, ENDSET
- Files used: Input on 3 or 5, output on 5.

Input Description: SET contains children set number.

Output Description: The parent set description in the next available set in the status table – the set is on unit 5.

Process Description:

1. Initialize I/O files, counters
2. Get next record ID from input, end
3. Error check
4. Form parent record ID by setting right half of ID to zero, write result on output unit, ignore duplicates
5. Go to 2
6. Close output file, update status table, TABNO, etc.

Name: GETQ (preprocessor step 3)

Purpose: To use Q card files (only) in setting transaction dates.

Linkage:

- Calling sequence: CALL GETQ (IPX)
- Common blocks used: UNITS, FLAGS, IMAGES, LINES
- Subroutines or functions used: ABORT, HDR, BLKLN, COLABL, DASHES, COUNT, PAGE
- Files used: Units 1,2,3,4,5,6,7,8

Input Description: The input parameter IPX is the LACIE Phase (2 or 3 in integer form)

Array IUN contains logical unit numbers for input and output files.

Subroutine IM1GET is called to get card images through array IM1.

Output Description: Page headers and the Q card image are written to the report file. The default transaction date from the Q card is saved in array IMQ.

Process Description: This routine follows two separate paths for the two LACIE phases. Different input files and output files are set up for the two phases. The first Q card from the Phase II file is saved to provide the transaction date for those cards in which it has not been punched. Duplicate Q cards are detected: an exact duplicate is allowed, but 2 different Q cards will cause step 3 to abort.

Name: GETSTR

Purpose: To provide a record to CREATE

Linkage:

- Calling sequence: CALL GETSTR(ACC,REC)
- Common blocks used: Unknown
- Subroutines or functions used: Unknown
- Files used: Unknown

Input Description:

Output Description: ACC,REC

Process Description: Upon return, ACC should contain the record number in ascending sequence, and REC should contain the record. The first word of REC should contain the record length, not counting itself. ACC = 0 signals end of file.

Name:	HDINIT (postprocessor)
Purpose:	Puts headers on output files
Linkage:	• Calling sequence: CALL HDINIT • Common blocks used: UNITS • Subroutines or functions used: • Files used: Units 1, 2, 3, 4, 5, 6, 7, 8
Input Description:	Logical unit numbers in array IUN
Output Description:	Appropriate header lines are written out to the output files
Process Description:	FORTRAN formatted write statements

Name:	HDR (preprocessor step 3)
Purpose:	Print the header label information at the top of a page.
Linkage:	• Calling sequence: CALL HDR • Common blocks used: LINES, UNITS, IMAGES • Subroutines or functions used: BLKLN • Files used: Logical UNIT = 7
Input Description:	Header date in IMQ array (Q card image)
Output Description:	LINE counter set to 12 Column label flag reset (ICOLAB = 0) Header printed on listing file (Unit 7)
Process Description:	Write the header and two blank lines, then set the line counter and column label flag.

Name: HEADER

Purpose: To print a line of text report. Provides for comments and headers.

Linkage:

- Calling sequence: CALL HEADER
- Common blocks used: SYSCOM, SY2COM
- Subroutines of functions used: INDEX, INPARM, SUBSTR
- Files used: Report file (12), message file (7) and command file (13).

Input Description: User's command string through common block SYSCOM (STR).

Output Description: One line of the header contents or comments is printed.

Process Description: The following processing is performed:

The command string is parsed.

Print header or comment up to 66 characters long as input if N = 1.

Read record line of header or comment up to 62 characters long from command file and print it's contents immediately after where first line ended if N = 2.

N should be either 1 or 2 syntax error is printed otherwise.

Name: IM1GET (preprocessor step 3)

Purpose: To check card images for duplicates and write them out.

Linkage:

- Calling sequence: flag = IM1GET (unit)
- Common blocks used: FLAGS, IMAGES, UNITS
- Subroutines or functions used: None
- Files used: UNIT IUNX, IU38

Input Description: Input update card file unit number in parameter unit and either (a) a card image already read into array IM2 or (b) flag IM2MT=1 when IM2 is empty because an end-of-file was read from the unit file. Unit number of error file for duplicates, IU38.

Output Description: The flag value from this function
- = 0 where a new card image is in IM1
- = 1 when no image is available

If available, one more image has been read into IM2 and checked for duplication of IM1. Duplicates are written to error file IU38. The next card image to be used is in IM1.

Process Description: Move card image 2 into card image 1 (if there is a card image in card image 2). Read another image (if any) into card image 2, checks the two images (card image 1 and card image 2) for duplicates. Duplicates are written to the error file and reading continues until a different card is

Process Description: (concluded)

encountered or the end of the input file is reached. At end-of-file, the flag IM2MT is set = 1 to indicate that array IM2 is empty.

Name: INIT (preprocessor step 1)

Purpose: Initializes step 1 output files.

Linkage:

- Calling sequence: CALL INIT
- Common blocks used: UNITS
- Subroutines or functions used: None
- Files used: Logical units 7, 8, and 9

Input Description: Array of logical unit numbers, IUN.

Output Description: Header lines for the listing files for invalid cards.

Process Description:

1. Set unit numbers to values from IUN
2. Write header records to those files

Name: IRECK (preprocessor step 5)

Purpose: To read input card images.

Linkage:

- Calling sequence: flag = IRECK(index)
- Common blocks used: CARDS, UNITS
- Subroutines or functions used: None
- Files used: Unit IUN(1), the input file

Input Description: Input parameter index specifies which of the four card buffers in array IM should receive the card image from the input file.

Output Description: The flag value of this function is =0 if an image was read, =1 after an end-of-file. The card image is left in the specified slot of array IM.

Process Description: Read the input file and set the flag.

Name: IUNLD

Purpose: To test for unload (UNLD) cards and reformat them into an "N" card.

Linkage:

- Calling sequence: CALL IUNLD (image)
- Common blocks used: None
- Subroutines or functions used: None
- Files used: None

Input Description: The input array image is an 80-character ASCII code card image.

Output Description: The array image is reformatted in place to the usual ASATS format.

Process Description: If the image does not match the string 'UNLD', return. If it does match, leave the 'N' in column 2, move the segment number from columns 10-13 to columns 4-7, move the acquisition date from 17-20 to 9-12 and insert a "3" in column 8. The tape number from columns 45-50 will be moved to columns 19-24. The transaction date field columns 14-17 will be left blank for insertion of the default date from the Q card, and the remaining unused columns will also be set blank.

Name: Update Processor Main Program

Purpose: To control execution of bulk update from ASATS cards.

Linkage:

- Execution: RUN ASATS
- Common blocks used: SYSCOM, PDT
- Subroutines or functions used: BE, REAF, CLOSEP, END, AUDATE, ASN, RED
- Files used: U(14) is used for reading commands.

Process Description: Reload and interprets commands. Causes appropriate subroutines to be executed to process given commands. Acceptable commands are: BE, RF, RE, UP, EN.

Name: KSGEQ (preprocessor step 5)

Purpose: To compare the segment number of the card just read to the first card in the buffer

Linkage:

- Calling sequence: flag = KSGEQ(index)
- Common blocks used: CARDS
- Subroutines or functions used: None
- Files used: None

Input Description: The input index specifies which card slot in array IM is to be compared to the card in the first slot. The two card images are both in array IM.

Output Description: The value of the output flag is =1 if the segment numbers are equal and =0 if they are different.

Process Description: Loop through the 4 segment number digits and set the flag to zero if they are different.

Name:	LAST
Purpose:	To find the last non-blank character in a string.
Linkage:	• Calling sequence: I = LAST (STRING, START, LENGTH) • Common blocks used: None • Subroutines or functions used: None • Files used: None
Input Description:	STRING = string name START = starting position LENGTH = length of substring
Output Description:	Location of last non-blank.
Process Description:	Establishes index pointer to last non-blank character in the string.

Name: MAIN (postprocessor)

Purpose: Gives a listing of new segment records, invalid new acquisitions, cards to be punched, a file for images of cards punched, updates for records, and packet labels for a RIMS update input file.

Linkage:

- Calling sequence: Not applicable
- Common blocks used: UNITS, ARRAYS
- Subroutines or functions used: UNINIT, HDINIT
- Files used: Units 5, 6, 8, 10

Input Description: Input file from RIMS update process. Each input record contains a carriage control in the first character and an ASCII digit designating the output file in the second character.

Output Description: The records from the input file are copied to the output file(s) designated by the second character of each record. (The second character itself is not written out).

Process Description: A header record is written to identify each of the output files. After copying all the input records to the appropriate output files, the choice of program STOP statements will give a message on the operator's console to let him know whether the punch and label files are empty for today's run.

Name: Main program, preprocessor step 1

Purpose: To control listing of update cards and the checking for invalid cards.

Linkage:

- Calling sequence: Not applicable
- Common blocks used: IMAGES, BCHK, HOLCON, UNITS, VALID, FLAGS
- Subroutines or functions used: ABORT, BLNKCK, CKTYLP, EXIT, INIT, UNINIT
- Files used: Logical units 1, 3, and 10

Input Description: ASATS update card images from logical unit 1

Output Description: Unsorted listing of input on unit 3, counts of valid and invalid cards on unit 10, error abort message on unit 10

Process Description: This program reads the update cards and lists them. It calls subroutines to check them for validity and write the valid cards to files for later processing.

Name: Main program (preprocessor step 3)

Purpose: Gives a report of the update card images. The report includes counts of each card and total counts for each phase.

Linkage:

- Calling sequence: Not applicable
- Common blocks used: UNITS, LINES, IMAGES, FLAGS, LEGTY
- Subroutines or functions used: UNINIT, GETQ, PHASE, HDR, COLABL, EQUALS, COUNTS, PAGE, EXIT, NONQ, ABORT, IMIGET, PAGE
- Files used: Logical UNITS = 1,2,3,4,5,6,7,8

Input Description: ASATS update card images

Output Description: Audit files for update cards and the update card files themselves.

Process Description: This program controls the sequence of calls to produce a combined report of update cards for Phase 2 and Phase 3, separated within the phases by card type. Page headers and page numbers and a grand total count of cards are given.

Name: MAIN (preprocessor step 5)

Purpose: To control the separation of complete segment descriptions (*, 2, and 3 cards for a segment), which may be either new segments or updates for existing segments, from single update cards.

Linkage:

- Calling sequence: Not applicable
- Common blocks used: CARDS, UNITS
- Subroutines or functions used: UNINIT, WRNUK, MVMR1, IRECK, KSGEQ
- Files used: Units 1, 3, 5, 7

Input Description: The input file (unit 1) contains all the *, 2, 3 cards in today's update deck sorted into order by segment so that all cards for one segment will be found in sequence.

Output Description: Output files units 3 and 5 contain, respectively, the complete sets, with the "*" card changed to a "Z" type, and the incomplete sets.

Process Description: Cards are read (by subroutine IRECK) into the 4-card buffer IM until either a change of segment number occurs, or the buffer is full, or the end-of-file is reached. A check is then made to see whether 3 cards for one segment of the three types *, 2, and 3 have been read. If so, they are saved as a set. Otherwise, they are saved as a nonset, and the process repeats.

Name: MOVSEG

Purpose: To parse the command for MO and to move information from the status table into a record.

Linkage:

- Calling sequence: CALL MOVSEG
- Common blocks used: SYSCOM
- Subroutines or functions used: GETREC, CHAR, REPR
- Files used: U(7)

Input Description: STR, the text of the command line

Output Description: Updated record in base.

Process Description: The set count from TAB is converted to character form and placed in the proper record.

Name: MVMR1 (preprocessor step 5)

Purpose: To move a line in an array to the beginning of the same array.

Linkage:

- Calling sequence: CALL MVMR1 (index)
- Common blocks used: CARDS
- Subroutines or functions used: None
- Files used: None

Input Description: Input parameter index specifies which card image slot in array IM should have its data moved up to the first slot.

Output Description: The card image in the first slot of IM.

Process Description: Loop through the image and move one character at a time.

Name: NONQ (preprocessor step 3)

Purpose: Puts the default value of transaction date from the Q card into those card images which need them.

Linkage:

- Calling sequence: flag = NONQ (unit)
- Common blocks used: FLAGS, IMAGES, LEGTY, UNITS
- Subroutines or functions used: IM1GET
- Files used: Update card input file (unit)

Input Description: An update card image in array IM1 is supplied by the call to IM1GET. Array IMQ contains the transaction date from the Q card.

Output Description: ASATS card image in IM1, with the transaction date inserted in those cards which need it. The flag value from NONQ is 0 when a new card is in IM1, 1 otherwise.

Process Description: Get a card image (and quit at end of file). If the card type field matches one of the list of card types in array IVTYP, the transaction date field is checked. If that field is blank, the default date from the Q card image is inserted into it.

Name: PAGE (processor step 3)

Purpose: Writes out a page number at the bottom of a page.

Linkage:

- Calling sequence: CALL PAGE
- Common blocks used: LINES, UNITS
- Subroutines or functions used: BLKLN
- Files used: UNIT = 7

Input Description: Logical unit number of the listing file in IU37

Current page count IPAGE

Current line number LINE

Output Description: Incremented page number in IPAGE.

On the listing file: enough blank lines to reach the bottom of the current page, and the page number

Process Description: The page number is negated to get a leading hyphen, and a trailing hyphen is supplied by the format.

Name: PARSEC

Purpose: To parse the command string for GC.

Linkage:

- Calling sequence: CALL PARSEC
- Common blocks used: SYSCOM
- Subroutines or functions used: GETCLD
- Files used:

Input Description: Command line in STR

Output Description: Set number in parameter K to GETCLD.

Process Description: Call INPARM to convert set number to binary, and call GETCLD.

Name:	PARSEP
Purpose:	To parse the command string for GP.
Linkage:	• Calling sequence: CALL PARSEP • Common blocks used: SYSCOM • Subroutines or functions used: GETPAR • Files used:
Input Description:	Command line in STR array
Output Description:	Set number in parameter K to GETPAR
Process Description:	Use INPARM to convert number, check it is legal set number, and call GETPAR

Name: PHASE (preprocessor step 3)

Purpose: Output all cards of all types found in one phase.

Linkage:

- Calling sequence: CALL PHASE (IP)
- Common blocks used: UNITS, LINES, FLAGS
- Subroutines or functions used: HDR, BLKLN, CARTP, PAGE, COLABL, DASHES, COUNT
- Files used: Logical UNIT = 7

Input Description: Input parameter IP in the LACIE phase (Integer 2 or 3)

Output Description: Count of cards for the phase in IPCT. Audit report on unit 7.

Process Description: This routine (by calling subroutines) formats a complete audit listing of all cards for one phase. It starts with a page header for the phase, steps through all card types in the file for that phase, and produces a total card count for that phase.

Name: PSWRD

Purpose: To read the password, scramble it and overstrike it on the scope.

Linkage:

- Calling sequence: CALL PSWRD (I)
- Common blocks used: SYSCOM
- Subroutines or functions used: None
- Files used: U(7), U(13)

Input Description: Password

Output Description: Scrambled password (in I)

NOTE: The scrambled password will be used as a record i.d. for a record containing the bit mask.

Process Description:

Name:	QCHCK (preprocessor step 1)
Purpose:	Check for duplicate or illegal Q card
Linkage:	• Calling sequence: CALL QCHCK • Common blocks used: FLAGS, IMAGES, UNITS • Subroutines or functions used: None • Files used: Units 4, 5, 7
Input Description:	Update card image in IMG array
Output Description:	A list of Q card images to an output file (units 4 and 5, respectively). A list of duplicate or different Q card images to unit 7 (if more than one Q card has inadvertently entered).
Process Description:	If counter IQCT=0, the first Q card is written to units 4 and 5, respectively. Then a check for duplicate or different Q cards is made. If a duplicate is found, the counter IQCT is left=1. If a different Q card is found, counter IQCT is incremented by 1 and an error flag is set.

Name: RED

Purpose: To read processing description for update card types and build processing description table.

Linkage:

- Calling sequence: CALL RED
- Common blocks used: /PDT/CTAB, FNO, RIDT, PTT
- Files used: Card images are read from unit 12.

Process Description: Card images are read until a blank card type is encountered. Table 7-3 illustrates format of cards. The results of the card images are stored in CTAB (card type table), FND (format number), RIDT (record ID type), and PTT (processing type table).

TABLE 7-3.– PROCESS DESCRIPTION
CARD IMAGE FORMAT

Field	Columns
Card Type	1
Input Format	2-4
Record ID Generation Type	5
Processing Category	6

Name: RESTRC

Purpose: To display a set of records containing information from both the records within a specified set and their parent records.

Linkage:

- Calling sequence: CALL RESTRC (SET)
- Common blocks used: SYSCOM, SY2COM
- Subroutines or functions used: DISFMT, INPARM, LMVTAB, LODFMT, LODREC, SETIN1, SUBSTR, TFORM, XXIN1
- Files used: None

Input Description: SET is the set number for a set of records with same format ID (i.e. a set of FLOCON records)

SY2COM contains the display format.

Output Description: Records within a given set and their parent records are displayed as specified.

Process Description: The following processing is performed.

(1) Get record ID from set.
(2) Retrieve record and associated format.
(3) Transfer data from retrieved record (FLOCON) to an output buffer according to the display format.
(4) Generate record ID for associated record with different format (DAPTS).
(5) Retrieve record with generated record ID and its's associated format.

Process Description: (6)Transfer data from retrieved record
(Continued) (DAPTS) to output buffer according to the display format.

(7)Display program (DISFMT) is called. (to write output buffer to report file).

Name: SMINUS

Purpose: To delete a password from the base.

Linkage:

- Calling sequence: CALL SMINUS
- Common blocks used: SYSCOM
- Subroutines of functions used: DECK,PSWRD
- Files used: u(7)

Input Description: PASSWORD

Output Description: Record removed from base.

Process Description: Hash the password, find it, and remove that record.

Name: SPLUS

Purpose: To add a password to the base.

Linkage:

- Calling sequence: CALL SPLUS(SECURF)
- Common blocks used: SYSCOM
- Subroutines or functions used: PSWRD,ADDR
- Files used: U(7), U(13)

Input Description: PASSWORD, BIT MASK

Output Description: New record in base.

Process Description: Hash the password and generate a new password record.

Name: STCNT

Purpose: To print a line of text followed by the number of entries in a given set. Provides for printing number of entries in a set with a label.

Linkage:

- Calling sequence: CALL STCNT
- Common blocks used: SYSCOM, SY2COM
- Subroutines of functions used: INDLX, INPARM, SUBSTR
- Files used: Report file (12), message file (7).

Input Description: User's command string including set number, column count and text comment through common block SYSCOM (STR).

Output Description: Text comment with number of entries is printed.

Process Description: The following processing is performed:

The command string is parsed.

A format is formed according to the given column count.

The text comment and number of entries are printed using the formed format.

The length of text comment is limited to (column count-4) characters long. Disregard any character exceeding the limit.

Name: TFORM (normal version)

Purpose: To transform data from one format to another format.

Linkage:

- Calling sequence: CALL TFORM(I)
- Common blocks used: SYSCOM,UPDCOM
- Subroutines or functions used: SUBSTR, VERIFY, INPARM, CHAR
- Files used:

Input Description:

- UPDCOM/BIOTAB, FLAG type of card being processed
- I is the source record
- /SY2COM/Buff contains input record
- /SY2COM/Buff contains output record
- /SY2COM/FMTID contains format ID's
- /SY2COM/length contains record lengths
- /SY2COM/MOVTAB contains pointers to fields of each record
- /SY2COM/NMOV is the number of fields to be transformed
- /SY2COM/FMT contains formats for both records

Output Description:

- SY2COM/buff contains the resulting record
- /UPDCOM/FLAG contains reject indicator

Process Description: The resulting record in the target buffer is constructed by moving data into a specified field. Data is moved according to the output type as follows:

- Output type 0 is a straight move of characters from the source buffer to the target buffer.
- Output type 9 moves data from the CC-ANCIL-TOPO table to the target buffer based on the source buffer value.
- Output type 5 moves data from the Film Products table to the source buffer value.
- Output type 4 moves data from the Computer Products table to the target buffer base on the source buffer value and the status (whether it is blank or not) of the "N" field.

Figure 7-1 depicts the program flow. Table 7-3 reflects the use of the film status table. Table 7-4 reflects the computer products status table. Table 7-5 reflects the CC-ANCIL-TOPO table.

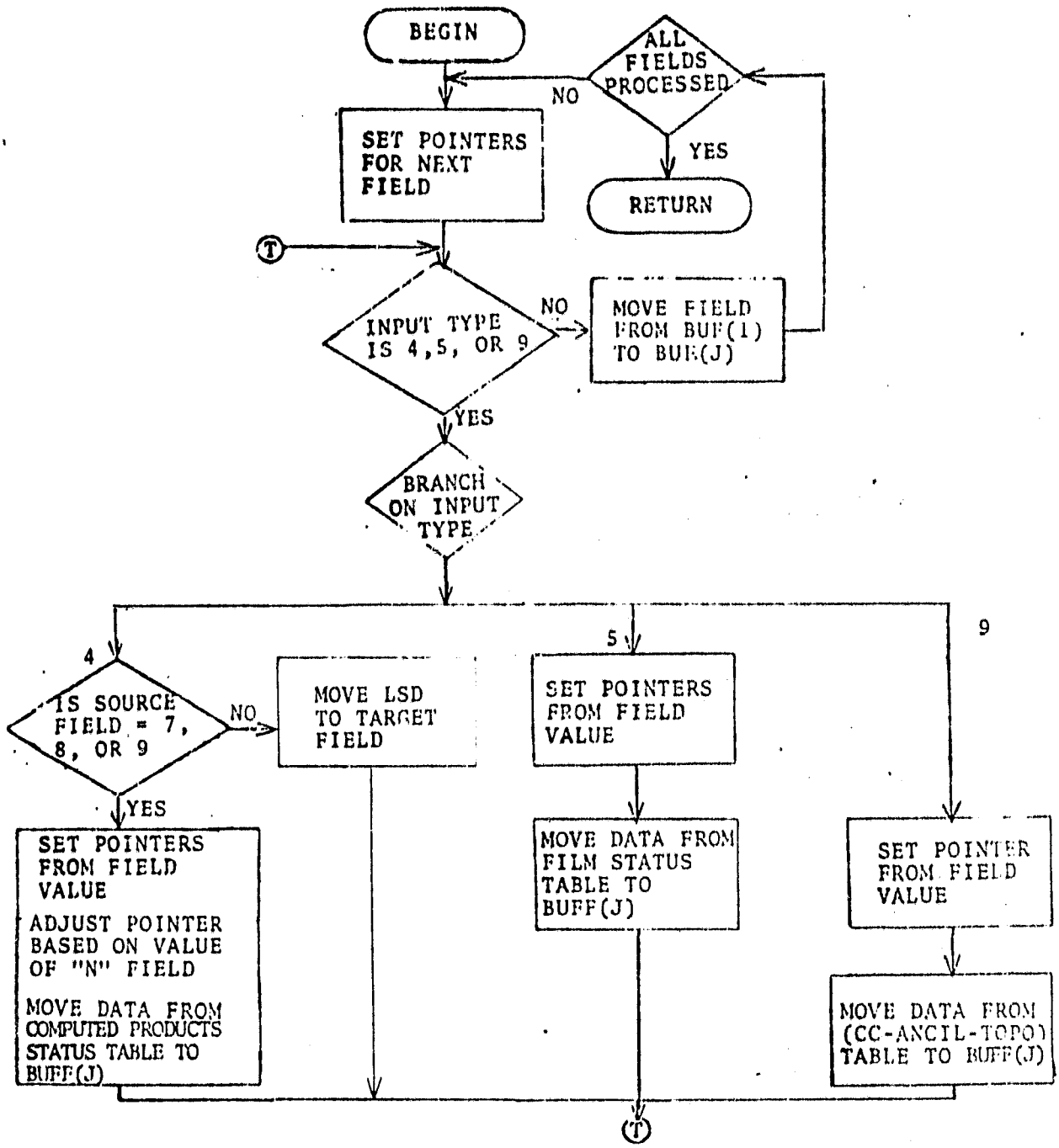


Figure 7-1.– TFORM for Normal RIMS.

TABLE 7-4.— FILM PRODUCTS STATUS TABLE (CURS1, output type 5 translation)

VALUE	MESSAGE
B	PFC WORK
G	LPDL REC'D
H	PKT AVAL
I	AI WORK
7	CANC
8	REOR
9	REJT

TABLE 7-5.— COMPUTER PRODUCTS STATUS TABLE
(CURS2, output type 4 translation)

VALUE	UNLOAD CONTENTS	MESSAGE
B	NA	C & I WORK
N	NA	I-100 RDY
J	NO	BATCH STD
J	YES	BATCH I-100
K	NO	ANAL STD
K	YES	ANAL I-100
M	NO	RERUN STD
M	YES	RERUN I-100
T	NA	I-100 ANAL
X	NA	Complete
7	NA	"LSD" Contents
8	NA	"LSD" Contents
9	NA	"LSD" Contents

TABLE 7-6.— CC-ANCIL-TOPO STATUS

VALUE	STATUS WORD
0	Await C/A/T
1	Await C/A
2	Await A/T
3	Await A
4	Await C/T
5	Await C
6	Await T
7	Complete

Name: TFORM (update version)

Purpose: To transform data from one format to another format.

Linkage:

- Calling sequence: CALL TFORM(I)
- Common blocks used: SYSCOM,UPDCOM
- Subroutines of functions used: SUBSTR, VERIFY, INPARM, CHAR
- Files used:

Input Description:

- UPDCOM/BIOTAB, FLAG type of card being processed
- I is the source record
- /SY2COM/Buff contains input record
- /SY2COM/Buff contains output record
- /SY2COM/FMTID contains format ID's
- /SY2COM/length contains record lengths
- /SY2COM/MOVTAB contains pointers to fields of each record
- /SY2COM/NMOV is the number of fields to be transformed
- /SY2COM/FMT contains formats for both records

Output Description:

- SY2COM/BUFF contains the resulting record
- /UPDCOM/FLAG contains reject indicator

Process Description: Fields are moved from the input record to the corresponding output record according to the input and output format specifications. Figure 7-2 reflects the detailed flow.

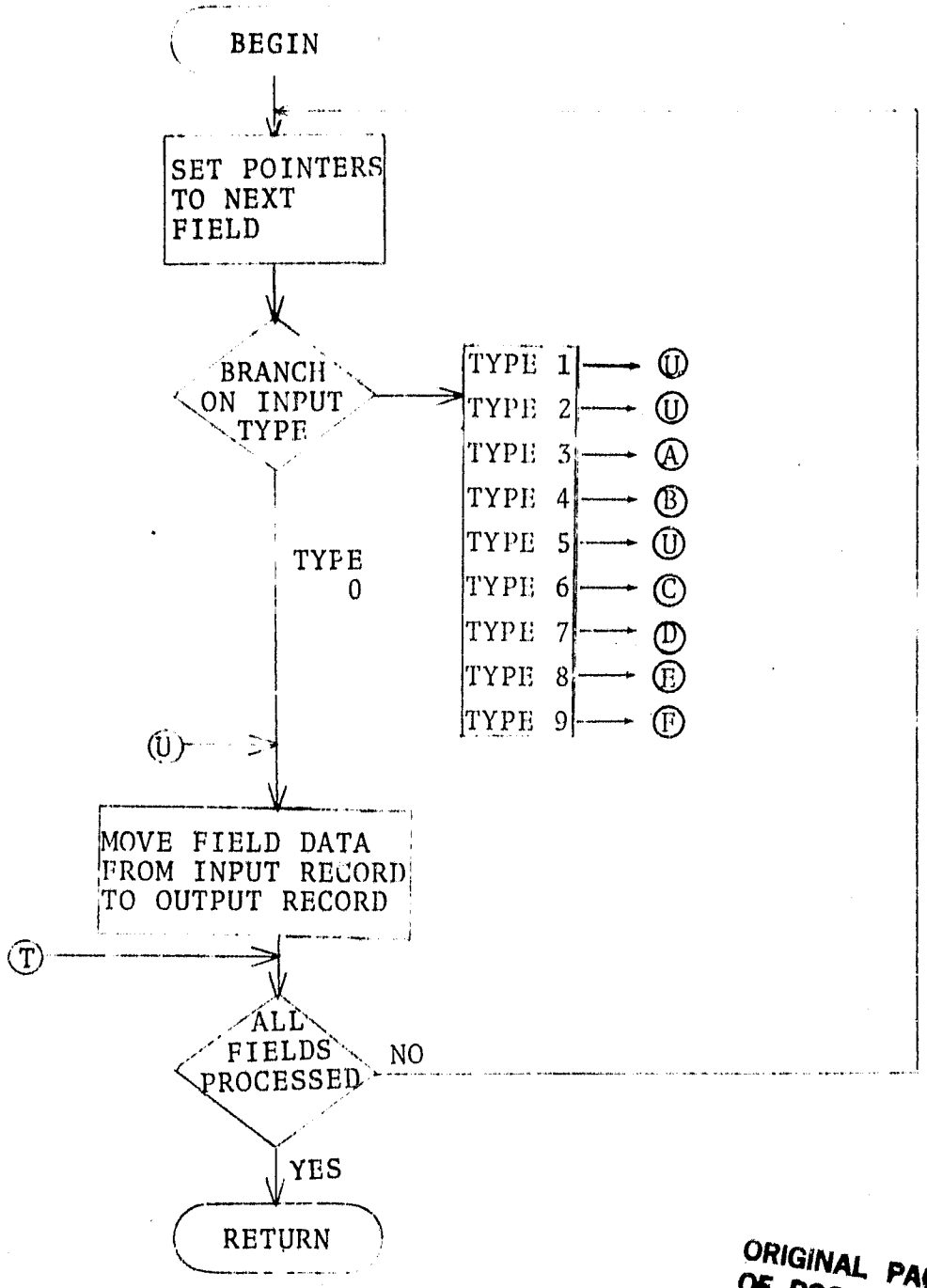


Figure 7-2.— TFORM for ASATS Update Processor.

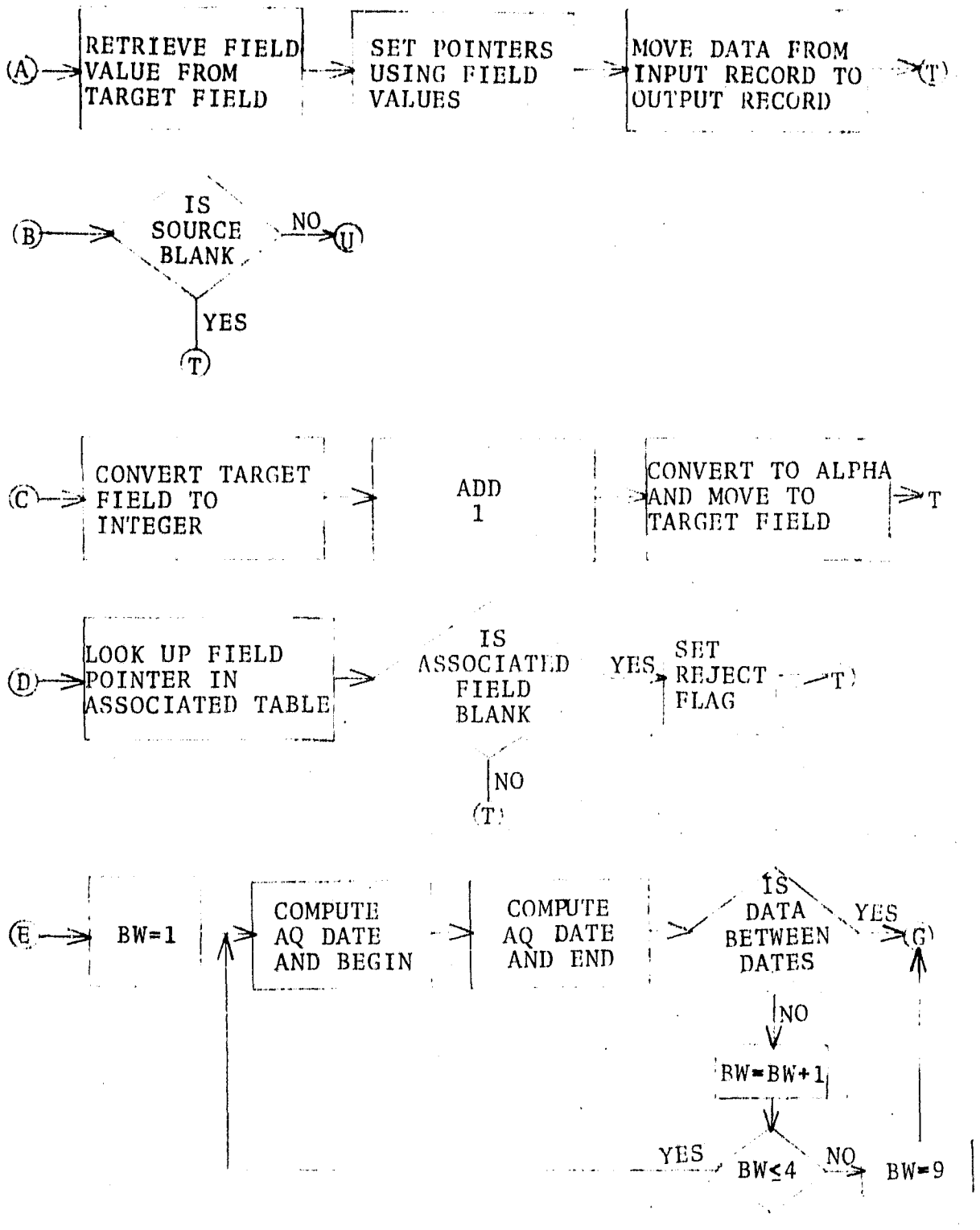


Figure 7-2.— Continued.

ORIGINAL PAGE IS
OF POOR QUALITY

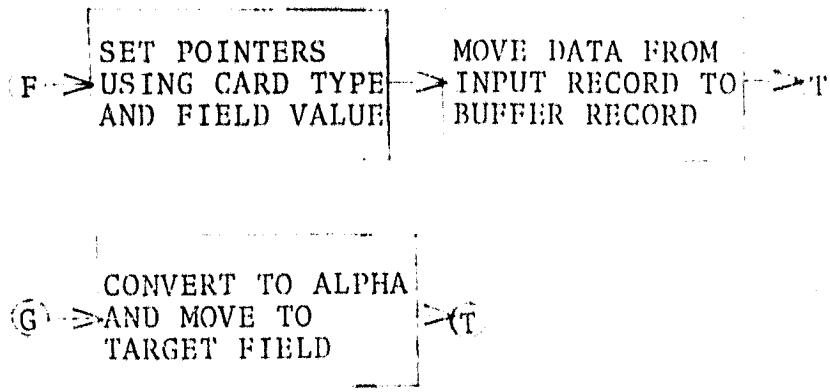


Figure 7-2.— Concluded.

Name: UNINIT

Purpose: Assigns logical unit numbers to file names at run time.

Linkage:

- Calling sequence: Call UNINIT
- Common blocks used: UNITS
- Subroutines or functions used: System subroutine ASSIGN
- Files used: STEP3.IUN as unit 20

Input Description: Reads 2-digit unit numbers from the first record with format 19I2. Rest of records to end-of-file are of the form 2I2, 15A2 and each contains a unit number, the number of characters in the file name, and the file name itself.

Output Description: Unit numbers are saved in array IUN. Unit-numbers-to-file-name associations are given to the system I/O control.

Process Description: Read the unit numbers and the file names and call the system routine ASSIGN.

NOTE: Slightly different versions of this routine are used by steps 1, 3, and 5 of the Preprocessor, by the Postprocessor, and by tasks JJ and CONTAP. The differences are just the name of the file containing the file names and the common block used to store unit numbers.

Name: UNLOCK

Purpose: To unlock the base.

Linkage:

- Calling sequence: CALL UNLOCK (SECURE)
- Common blocks used: SYSCOM
- Subroutines or functions used: PSWRD, LODREC
- Files used:

Input Description: Password

Output Description: Bit mask in SECURE

Process Description: The bit mask is used to lock (0) or unlock (1) the command. See JLASYS of the RIMS Maintenance Document for the command list.

Name: WRNUK (preprocessor step 5)

Purpose: To write lines of an array to an output unit.

Linkage:

- Calling sequence: CALL WRNUK (number, unit)
- Common blocks used: CARDS, UNITS
- Subroutines or functions used: None
- Files used: Output file unit selected by calling program.

Input Description: number card images in array IM, and the unit number for the output file.

Output Description: The card images for a set or for a partial set are written to the appropriate file.

Process Description: A FORTRAN formatted write statement is executed the specified number of times.